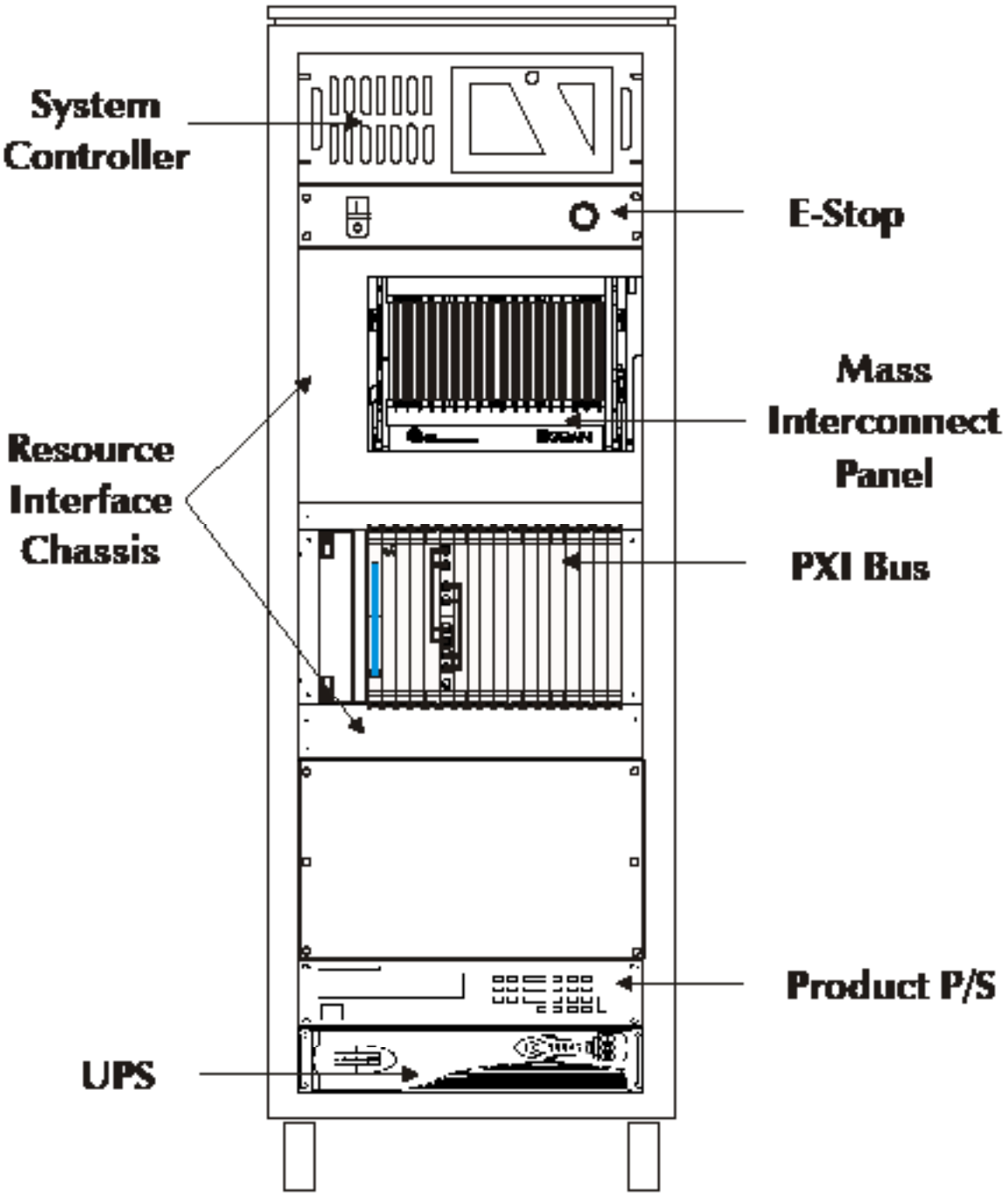
2050
User
Manual
4200-0339
V1.0

Table of Contents

Overview

Typical Series 2050 Test System



Overview

System Controller

The 2050 Series Test System uses a PC-based system controller and standard Windows® software operating system, supporting:

- GPIB
- Ethernet
- Environmental Chamber Control
- Process Control, including Handlers
- PXI Control
- Standard PC Peripheral and I/O Features

PXI Chassis and Instrumentation

The Series 2050 Test System is centered on the widely-supported PXI (PCI Extensions for Instrumentation) standard. The Series 2050 integrates an 18-slot 6U PXI Chassis that supports 3U instruments, such as any of the Digalog designed PXI cards or PXI instruments from any of over 60 third party venders who support the open-architecture PXI standard.

Resource Interface Chassis and Instrumentation

The Series 2050 is designed to take advantage of open-architecture bus technologies, while providing for the needs of a fully integrated test system using the Resource Interface Chassis (RIC).

For application specific needs, the RIC can be configured with a wide selection of signal conditioning, source, measurement, and switching capabilities. This configuration provides the accuracy required for many of today's test solutions.

Mass Interconnect

The Series 2050 utilizes the industry standard Virginia Panel Interface as the mass interconnect to the UUT. It can be configured with the exact mixture of high-density, high-current, coaxial connections needed for any particular UUT. This interface may be populated as required, and expanded to support changing needs. The Virginia Panel Interface provides a reliable, industry accepted adapter to the UUT, whether using a fixture or wiring harness.

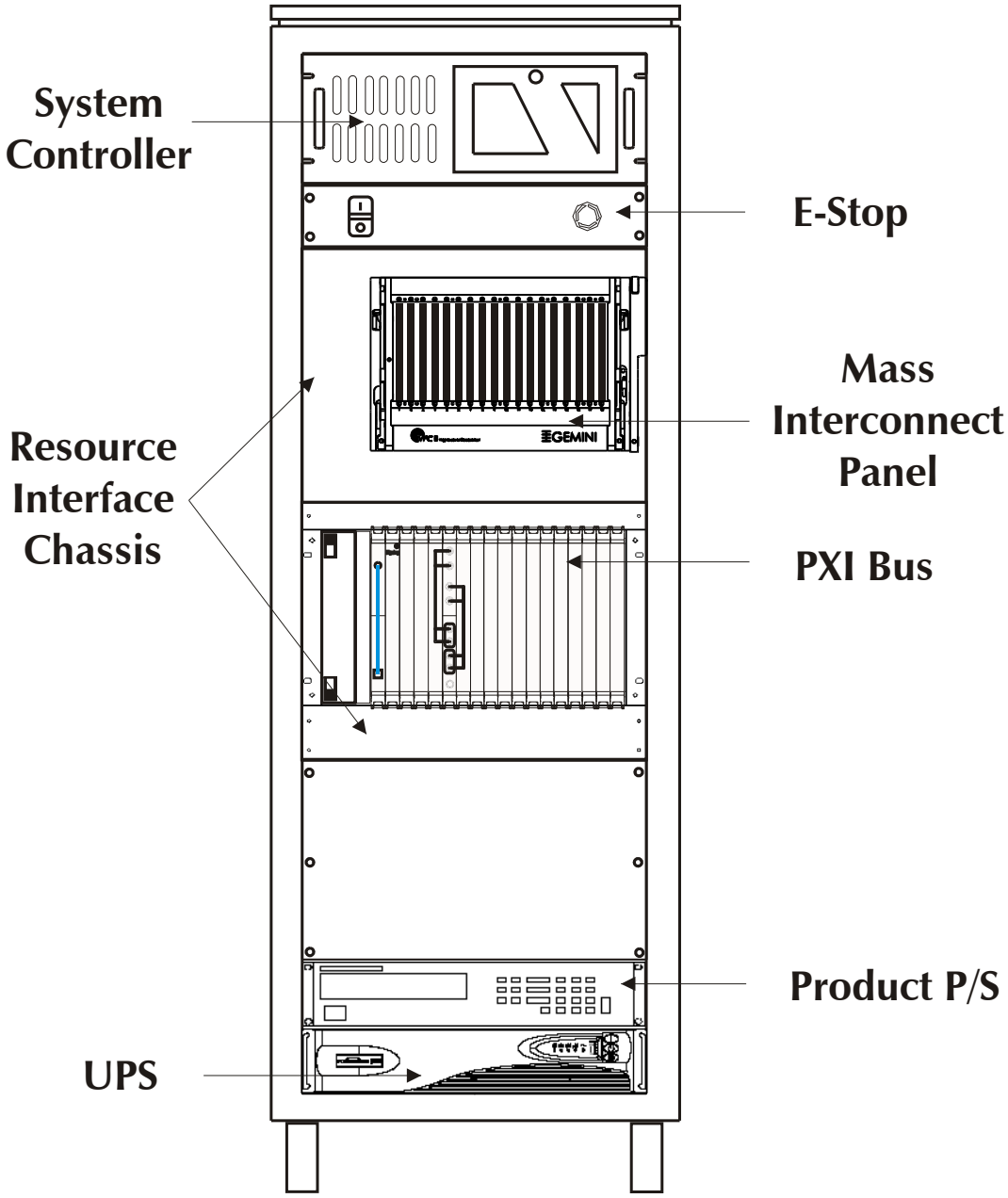
AC Power Management

The Series 2050 integrated AC power management assembly features include:

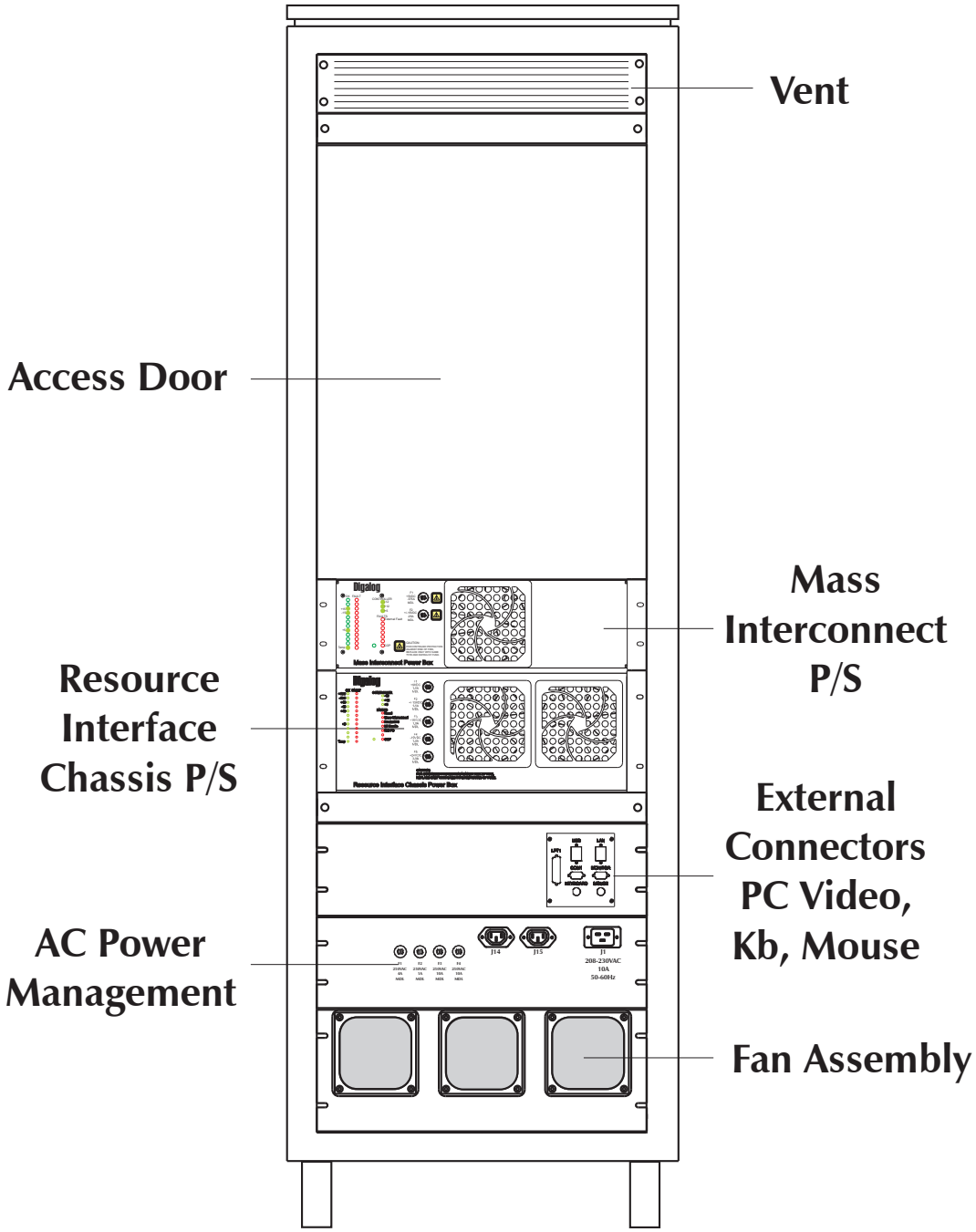
- Emergency Stop
- Uninterruptable Power Source
- Power Monitoring and Shutdown
- Power Distribution

Hardware

Typical 2050 Test System - Front

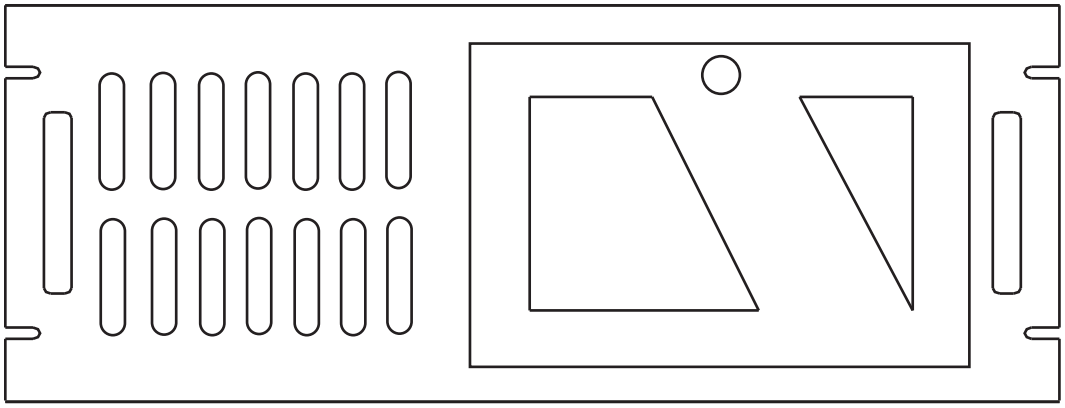


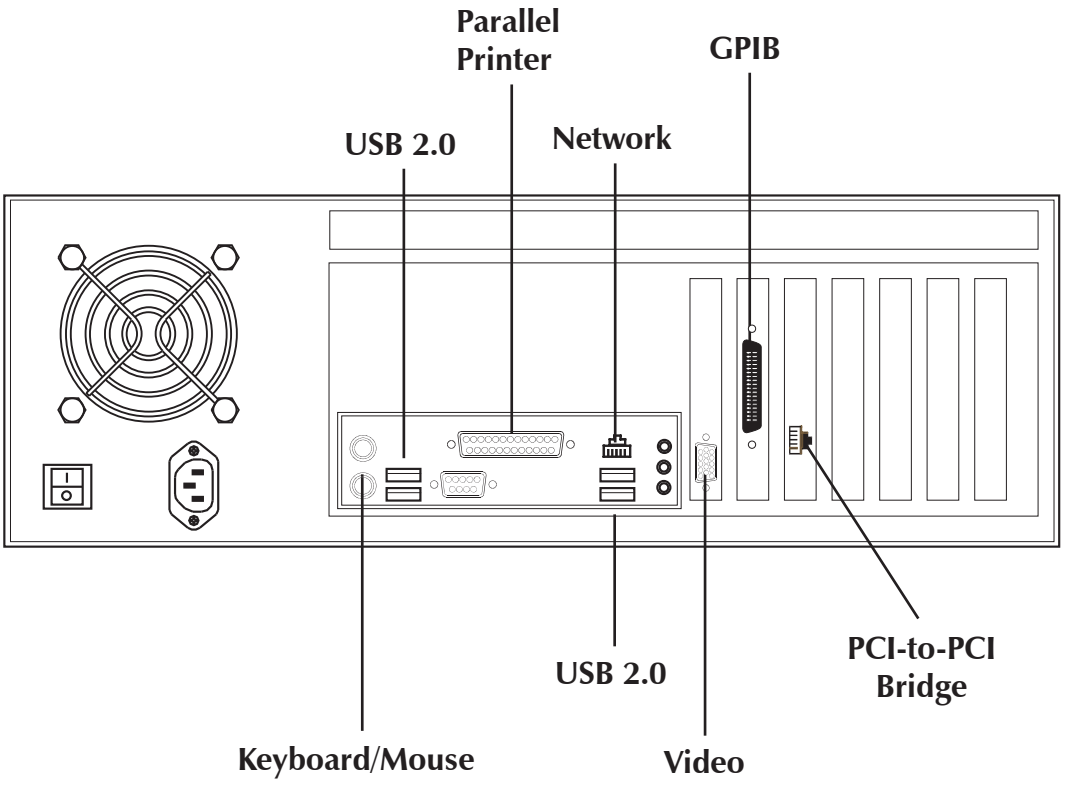
Typical 2050 Test System - Rear



System Controller

Typical 2050 System Controller





System Controller

The computer in the Series 2050 Test System is an industrial rack mount controller with an ATX form factor motherboard. It typically features:

- 2.8GHz Pentium IV processor, 533MHz FSB
- 1GB PC2700 333MHz DDR SDRAM
- 10/100 LAN
- Dual EIDE ATA100 ports
- On-board VGA & sound
- 4 USB 2.0 ports
- 1.44 MB floppy drive
- 40 GB ATA100 EIDE hard drive
- 4.7GB 4X DVD-RW, -R, CD-RW drive
- Windows XP Pro
- Norton AntiVirus
- Symantec V2i Protector

Serial Interface

The serial interface provides the communications link between the PXI rack and the computer. A PCI board is installed in the PCI bus of the system controller. A PXI board is also installed in the PXI rack. The PCI-to-PCI bridge uses a 2 meter copper cable to provide a high-speed 1.5Gb/s serial link.

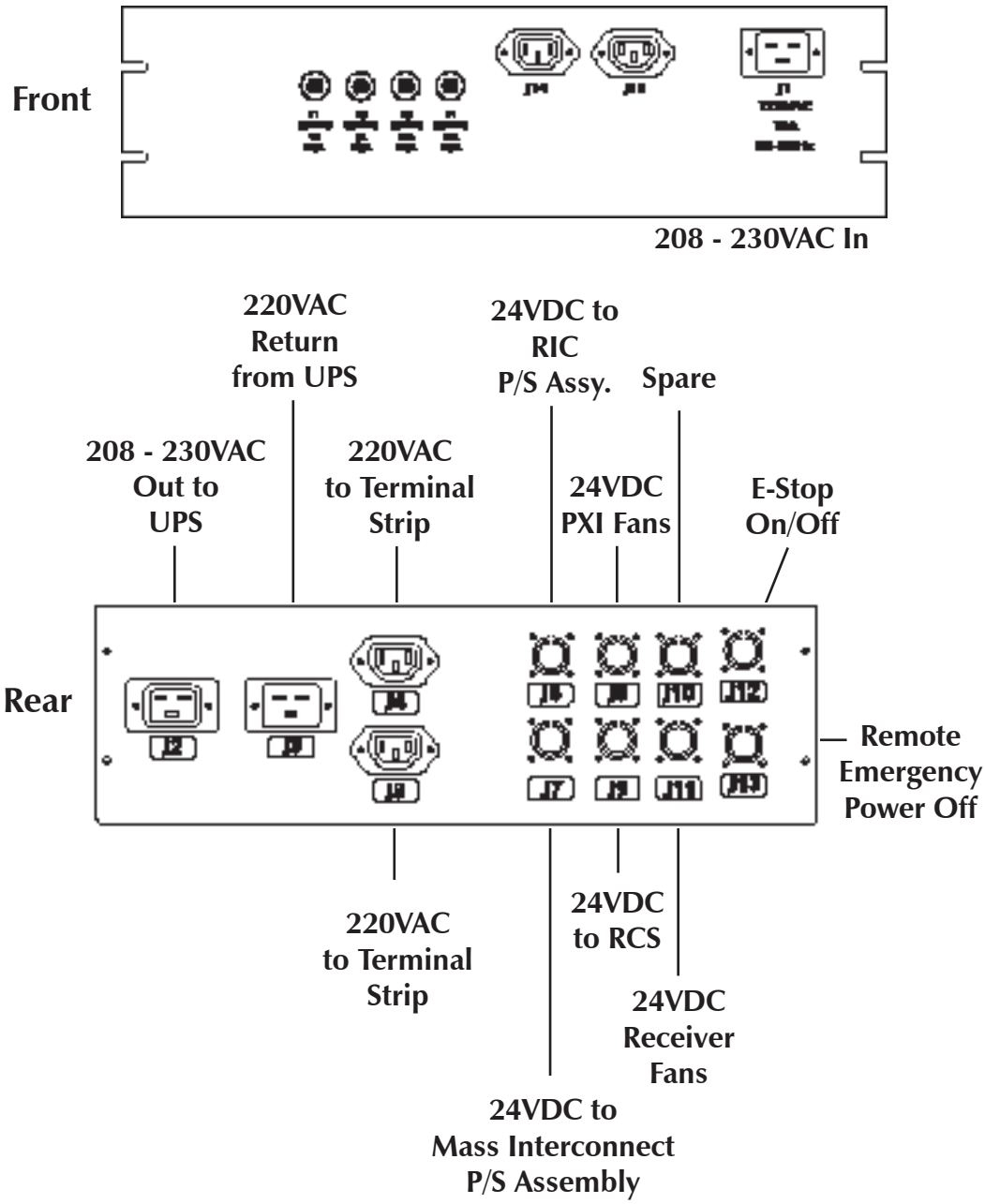
GPIB Communication Card

There is one PCI-GPIB (IEEE-488 card) installed in the PCI bus of the system controller. This device is used to communicate with the Resource Interface Chassis (RIC) Power supply assembly, the Mass Interconnect Power Supply, and the programmable Product Power Supply.

AC Power Management

AC Power Management

#0050-5100



AC Power Management Assembly - #0050-5100

The AC Power Management Assembly accepts 208 - 230VAC 50/60 cycle input voltage from the J1 receptacle on its front panel and routes it through the master relay to J2 on the rear panel. From there, it is routed to the the UPS. If the E-Stop switch is pressed, AC power is removed from J2 and the UPS is signaled to shut down.

Conditioned power is then routed back from the UPS to the AC Power Management assembly through connector J3. When the On/Off switch is closed, the 220VAC is connected to outputs J4 and J5 for use by the two AC Power terminal strips. These strips feed the UUT P/S, Mass Interconnect P/S, RIC P/S, System Controller, AC Fan Assembly, etc. Also when the On/Off switch is closed, the 220VAC is fed to the primary of a 24VDC linear power supply. This 24VDC is used by the control circuitry of the Mass Interconnect P/S, RIC P/S, Relay Control System (RCS), and for both RIC Fan Assemblies.

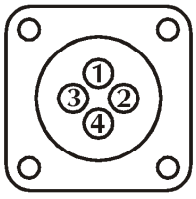
Another function of the 24VDC circuitry in the AC Power Management assembly is to sense the position of the E-Stop switch and shut down the UPS using the Remote Emergency Power Off (REPO) line when the switch is tripped.

Fuses

F1	4A, 250VAC input to 24VDC Power Supply
F2	1A for 24VDC out to connectors J6, J7, J8, J9, J10, & J11
F3	10A, 250VAC for connectors J4 & J5
F4	10A, 250VAC for connectors J14 & J15

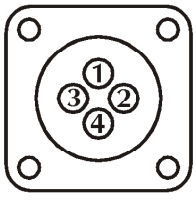
J6 - 24VDC Out to Mass Interconnect P/S

Connector	Signal
1	24VDC
2	N/C
3	GND
4	N/C



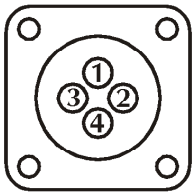
J7 - 24VDC Out to Resource Interface Chassis P/S

Connector	Signal
1	24VDC
2	N/C
3	GND
4	N/C



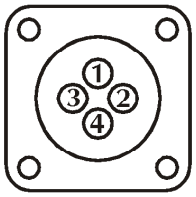
J8 - 24VDC Out to PXI Fan Assembly

Connector	Signal
1	24VDC
2	N/C
3	GND
4	N/C



J9 - 24VDC Out to RCS Controller

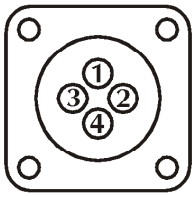
Connector	Signal
1	24VDC
2	N/C
3	GND
4	N/C



J10 - 24VDC Out - Spare

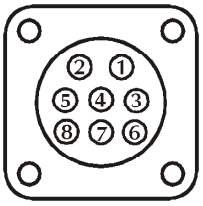
J11 - 24VDC Out to Receiver Fan Assembly

Connector	Signal
1	24VDC
2	N/C
3	GND
4	N/C



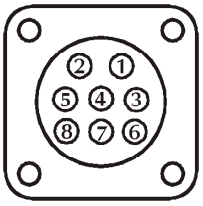
J14 - Out to E-Stop - On/Off Switch Assembly

Connector	Signal
1	EStop NC-1
2	EStop NC-2
3	EStop NO-1
4	EStop NO-1
5	Power Off
6	Power On/Off
7	Power On
8	Power Light Rtn.



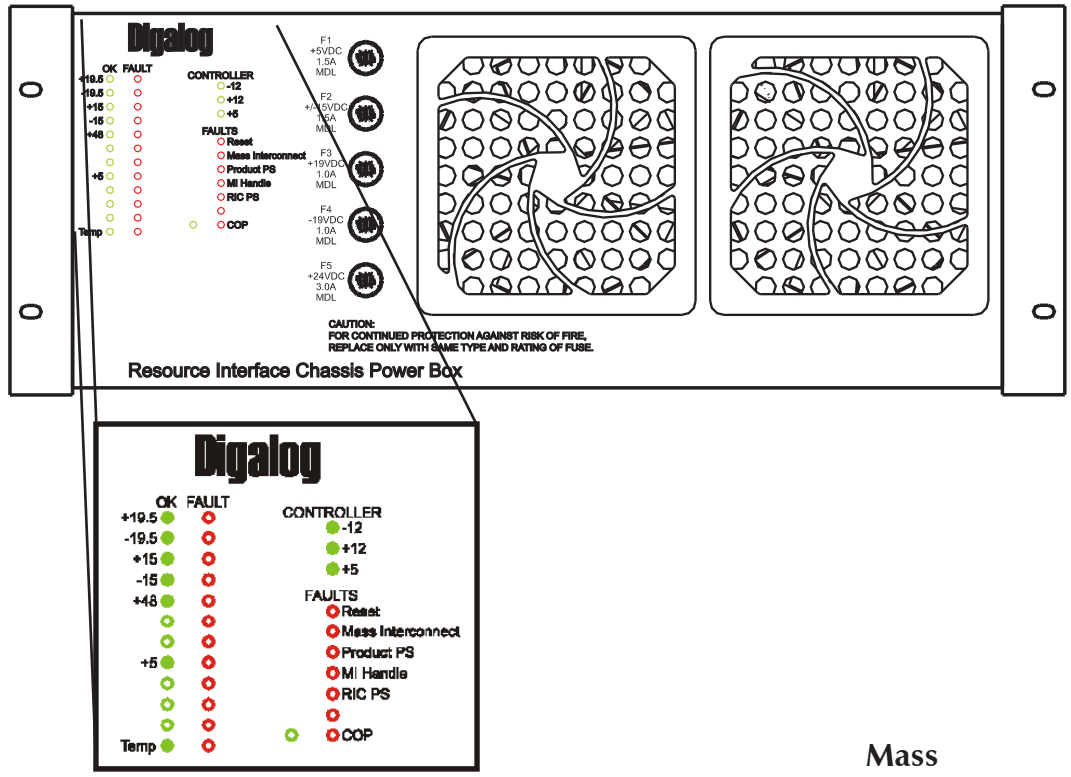
J15 - Out to REPO Signal on UPS

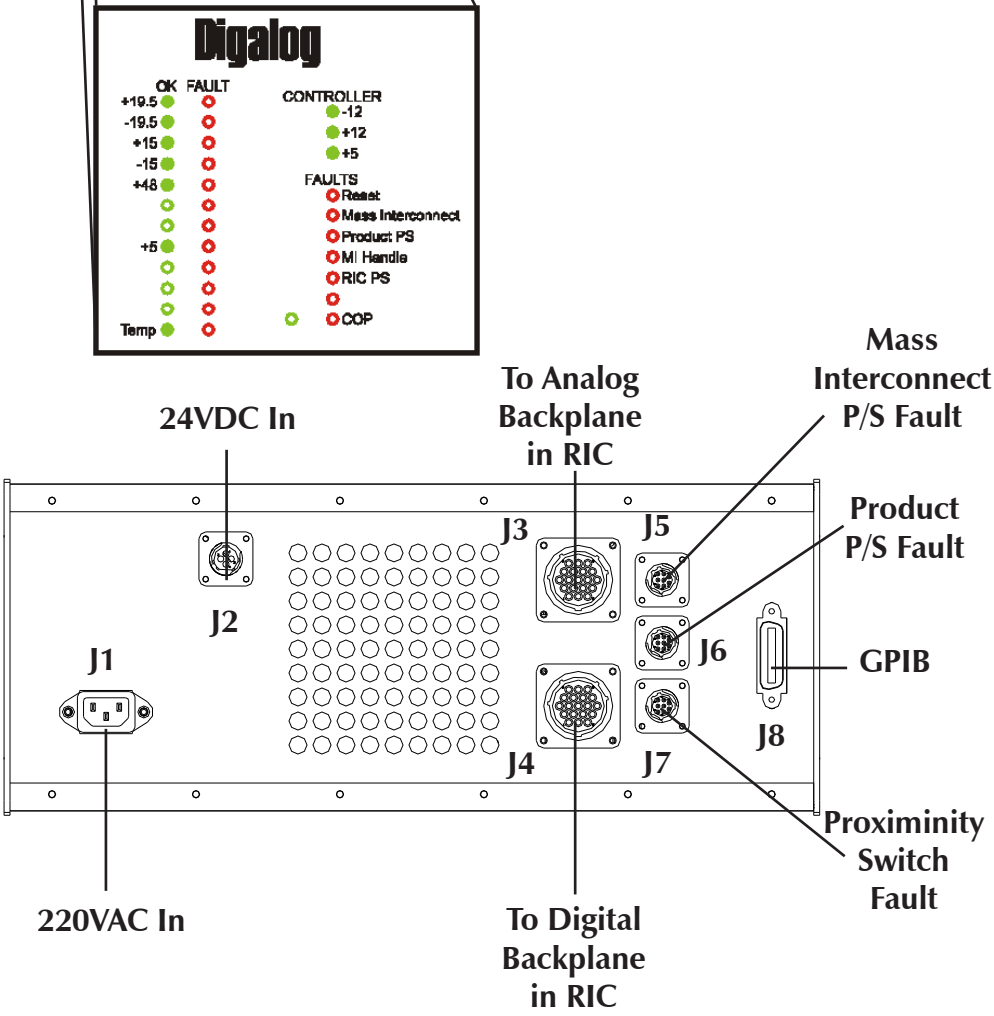
Connector	Signal
1	N/C
2	N/C
3	To REPO P1
4	To REPO P2
5	N/C
6	N/C
7	N/C
8	N/C



Resource Interface Chassis P/S

RIC P/S


Digalog
OK FAULT
+19.5
-19.5
+15
-15
+48
+5
Temp
CONTROLLER
-12
+12
+5
FAULTS
Reset
Mass Interconnect
Product PS
MI Handle
RIC PS
COP
F1 +5VDC 1.5A MDL
F2 +/-15VDC 1.5A MDL
F3 +19VDC 1.0A MDL
F4 -19VDC 1.0A MDL
F5 +24VDC 3.0A MDL
CAUTION:
FOR CONTINUED PROTECTION AGAINST RISK OF FIRE,
REPLACE ONLY WITH SAME TYPE AND RATING OF FUSE.
Resource Interface Chassis Power Box


24VDC In
To Analog Backplane in RIC
Mass Interconnect P/S Fault
Product P/S Fault
GPIB
Proximinity Switch Fault
J1
J2
J3
J4
J5
J6
J7
J8
220VAC In
To Digital Backplane in RIC

RIC Power Supply Assembly

The RIC Power Supply Assembly provides the 2050 Series Test System with ±15VDC @ 3.0A,+5VDC @ 18A, +19.5VDC @ 4.8A, and -19.5VDC @ 4.8A. The layout of the back panel is shown to the left. The following pages show the pinouts for both the J3 (Analog Backplane) and J4 (Digital Backplane) connectors. Note that the five power supplies utilize sense circuitry to maintain accurate output voltages with respect to the AC line voltage.

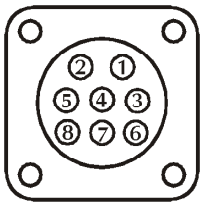
The use of a GPIB controller provides AC control, voltage monitoring, and fault loop control. The front panel contains a series of LED output indicators for monitoring the status of this assembly and the Mass Interconnect Power Supply assembly as shown to the left. The supply is cooled using positive fan assisted air flow through the use of two front mounted fans with exhaust air directed towards the rear of the assembly.

The connectors labeled "J5", "J6", and "J7" contain remote fault loop signals that are monitored and controlled by the RIC P/S controller board.

F1	**+5VDC**	**1.5A**
F2	**±15VDC**	**1.5A**
F3	**-19.5VDC**	**1.0A**
F4	**+19.5VDC**	**1.0A**
F5	**+24VDC**	**3.0A**

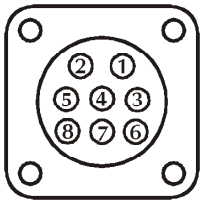
J5 - Fault 1

Connector	Signal
1	EXT_FAULT_1
2	GND
3	KEY PLUG
4	N/C
5	KEY PLUG
6	N/C
7	FAULT_OUT1_COL
8	FAULT_OUT1_EMTR



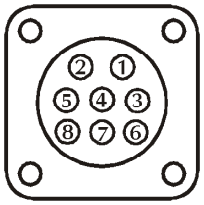
J6 - Fault 2

Connector	Signal
1	EXT_FAULT_2
2	GND
3	KEY PLUG
4	N/C
5	FAULT_OUT2_COL
6	FAULT_OUT2_EMTR
7	KEY PLUG
8	N/C



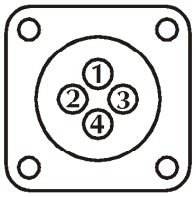
J7 - Fault 3

Connector	Signal
1	EXT_FAULT_3
2	GND
3	FAULT_OUT3_COL
4	KEY PLUG
5	FAULT_OUT3_EMTR
6	N/C
7	KEY PLUG
8	N/C

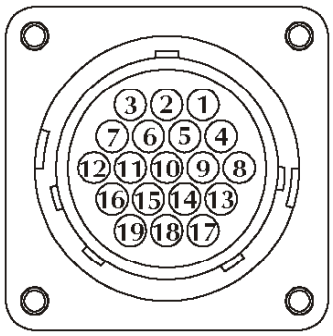


J2 - 24VDC In from AC Power Management Assembly

Connector	Signal
1	24VDC
2	N/C
3	GND
4	N/C

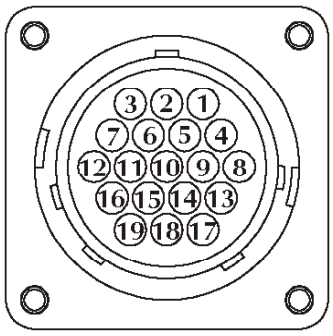


J3 - Analog Motherboard



Connector	Signal
1	+15V Sense +
2	+15V
3	+15V Sense -
4	+15V Common
5	+19.5V Sense +
6	+19.5VDC
7	+19.5V Sense -
8	+19.5V Common
9	-19.5V Sense - (-sense)
10	-19.5VDC (-out)
11	-19.5V Sense+ (+sense)
12	-19.5V Common (+out)
13	-15V Sense - (-out)
14	-15V (-out)
15	-15V Sense + (+sense)
16	-15V Common (+out)
17	N/C
18	N/C
19	KEY PLUG

J4 - Digital Motherboard

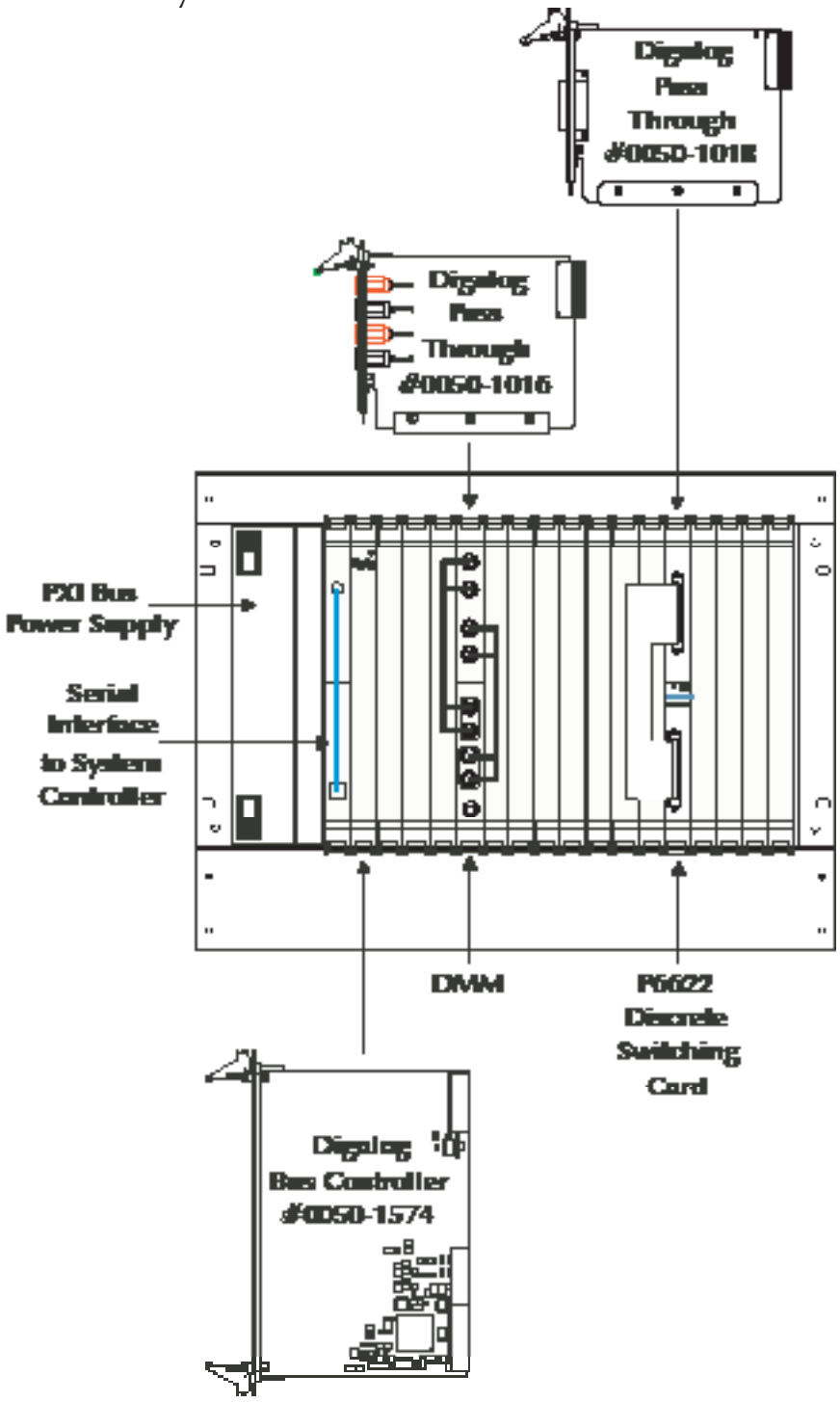


Connector	Signal
1	+5V
2	+5V
3	+5V Sense +
4	N/C
5	+5V Common
6	+5V Common
7	+5V Sense -
8	Keying Plug
9	N/C
10	EXT_PS_FAULT_LOOP_OUT
11	PS1_FAULT_COL
12	PS1_FAULT_EMTR
13	GND
14	GND
15	Keying Plug
16	N/c
17	N/C
18	N/C
19	N/C

PXI Chassis Assembly

PXI Chassis Assembly Typical

Typical
Switchir
DMM
Digitize
ARB



PXI Chassis Assembly

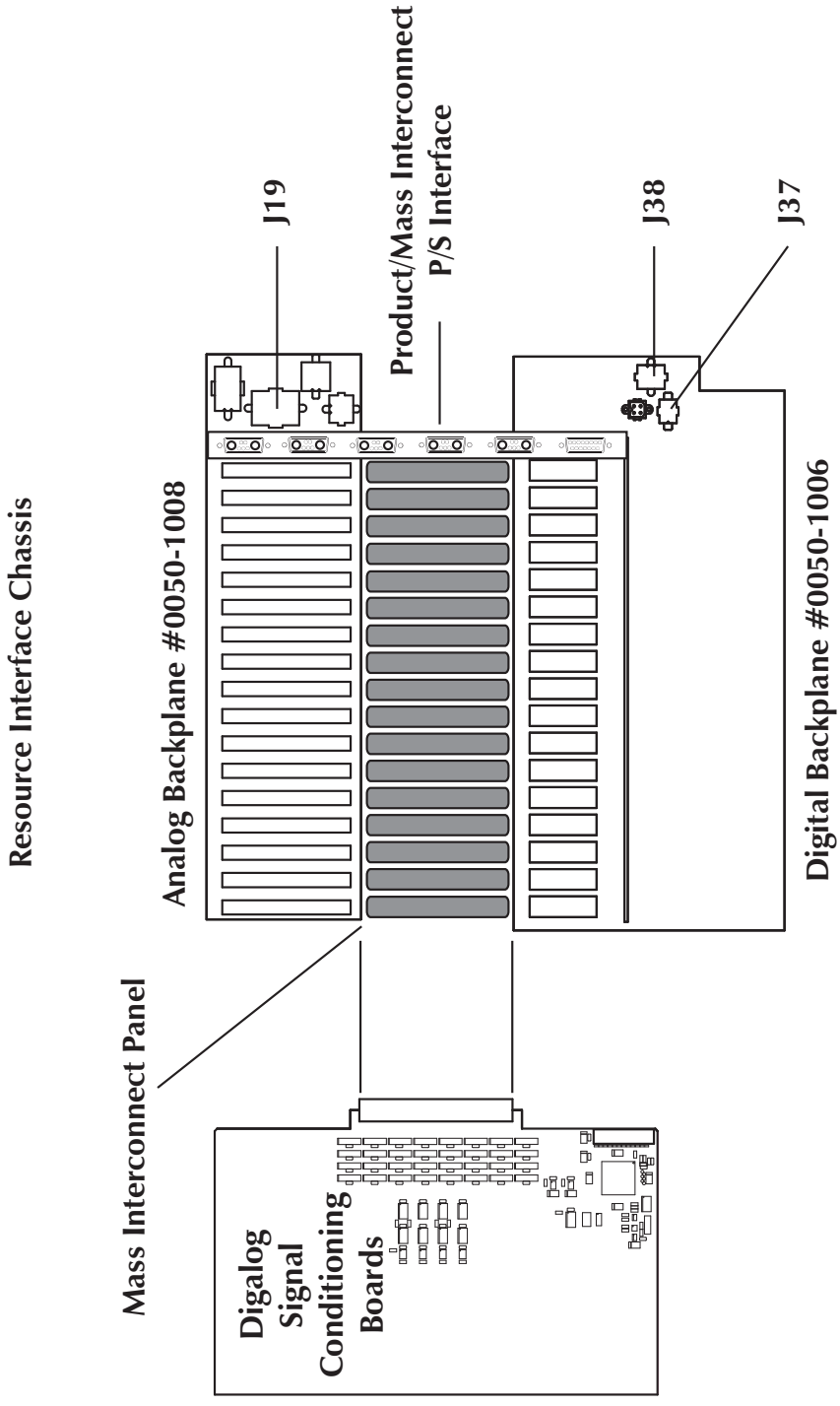
The PXI Chassis Assembly provides an open architecture solution for the 2050 Test System. It uses the industry standard PXI bus as the control for the instrumentation cards. It comes standard with a PXI power supply from Power One. By using a 6U chassis instead of a 3U chassis, there is enough room for the additionof a Digital Backplane for the Resource Interface Chassis (RIC).

The serial interface provides the communications link between the PXI rack and the system controller. A PCI board is installed in the PCI bus of the system controller. A PXI board is also installed in the PXI rack. The PCI-to-PCI bridge uses a 2 meter copper cable to provide a high-speed serial link.

An optional DMM can be used for measuring analog signals. The Pass through card acts as a 3U to 6U slot adaptor allowing a 3U card to be easily inserted into the 6U rack. The external signals from the DMM are connected to the Pass Through board through the use of a short external cable. These external signals are routed through the Pass Through board to the Digital Backplane on the RIC. Access to these external signals at the receiver panel is then possible if a 2050 board with signal pass through capabilities is present in the corresponding 2050 RIC slot.

The Bus Controller is a 6U PXI card designed to bridge/map 2050 card I/O logic into the system controller's memory for direct control. The design consists of a FPGA/PCI interface core, system board parallel bridge I/O control logic, system board serial I/O control logic, and PXI trigger bus I/O buffers. The FPGA and PCI core provides a compliant PCI/PXI interface. The custom control logic bridges I/O functions between the PCI/PXI bus and the 2050 RIC boards.

Resource Interface Chassis


Resource Interface Chassis
Mass Interconnect Panel
Analog Backplane #0050-1008
Digalog Signal Conditioning Boards
Product/Mass Interconnect P/S Interface
J19
J38
J37
Digital Backplane #0050-1006

Resource Interface Chassis

The Resource Interface Chassis (RIC) was designed to mate the resources of the PXI chassis to the Virginia Panel Interface and ultimately to the Device-Under-Test (DUT). It consists of a Digital Backplane for communication with the PXI Chassis Assembly and an optional 2050 system board control. The Analog Backplane is used for switching and signal conditioning to/from the Virginia Panel.

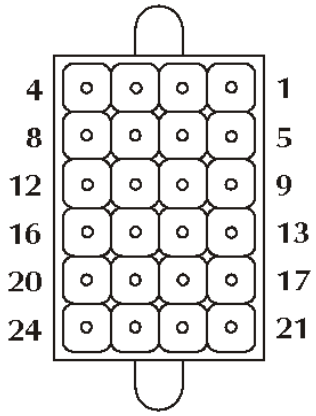
The Mass Interconnect Panel itself contains seventeen (17) connectors, each with 192 pins per connector for a 3264 pin patch panel. It also has a positive locking mechanism for the test fixture and a proximity switch for fault control.

Power to the Mass Interface Panel is provided by the Product/mass Interconnect P/S Interface, #0050-5104. This assembly can handle the Mass Interconnect P/S outputs and up to five Product Power Supplies.

Pinouts for the power supply connectors is as follows:

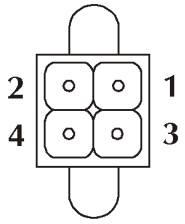
J19 - Analog Motherboard

Connector	Signal
1	+15V Sense+
2	+15VDC
3	+15V Sense-
4	+15V Common
5	+19.5V Sense+
6	+19.5VDC
7	+19.5V Sense -
8	+19.5V Common
9	N/C
10	N/C
11	N/C
12	N/C
13	N/C
14	N/C
15	N/C
16	N/C
17	-19.5V Sense- (-Sense)
18	-19.5VDC
19	-19.5V Sense+ (+Sense)
20	-19.5V Common (+Out)
21	-15V Sense (-Sense)
22	-15VDC
23	-15V Sense+ (+Sense)
24	-15V Common (+Out)



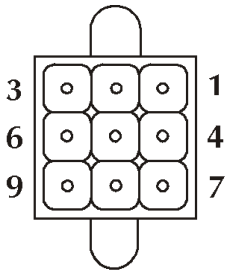
J37 - Digital Motherboard

Connector	Signal
1	+48V Sense+
2	+48VDC
3	+48V Common
4	+48V Sense-



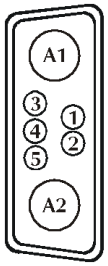
J38 - Digital Motherboard

Connector	Signal
1	+5VDC
2	+5VDC
3	+5V Sense+
4	PS1_FAULT_COL
5	EXT_PS FAULT LOOP OUT
6	PS1_FAULT_EMTR/GND
7	+5V Sense-
8	+5V Common
9	+5V Common



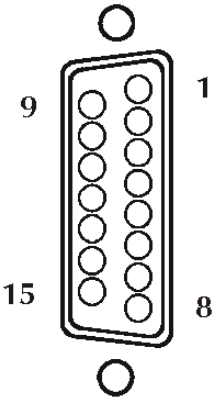
PS0 - PS5 - Product/Mass Interconnect P/S Interface

Connector	Signal
A1	+Out1, +Out2
A2	-Out1, -Out2
1	N/C
2	N/C
3	+Sense
4	N/C
5	-Sense



PBPS - Product/Mass Interconnect P/S Interface

Connector	Signal
1	+5VDC
2	+5V Sense-
3	N/C
4	+15V Sense+
5	+15V Sense-
6	N/C
7	-15V Sense+
8	-15V Sense-
9	+5V Sense+
10	+5V Common
11	+15VDC
12	+15V Common
13	N/C
14	-15V Common
15	-15VDC

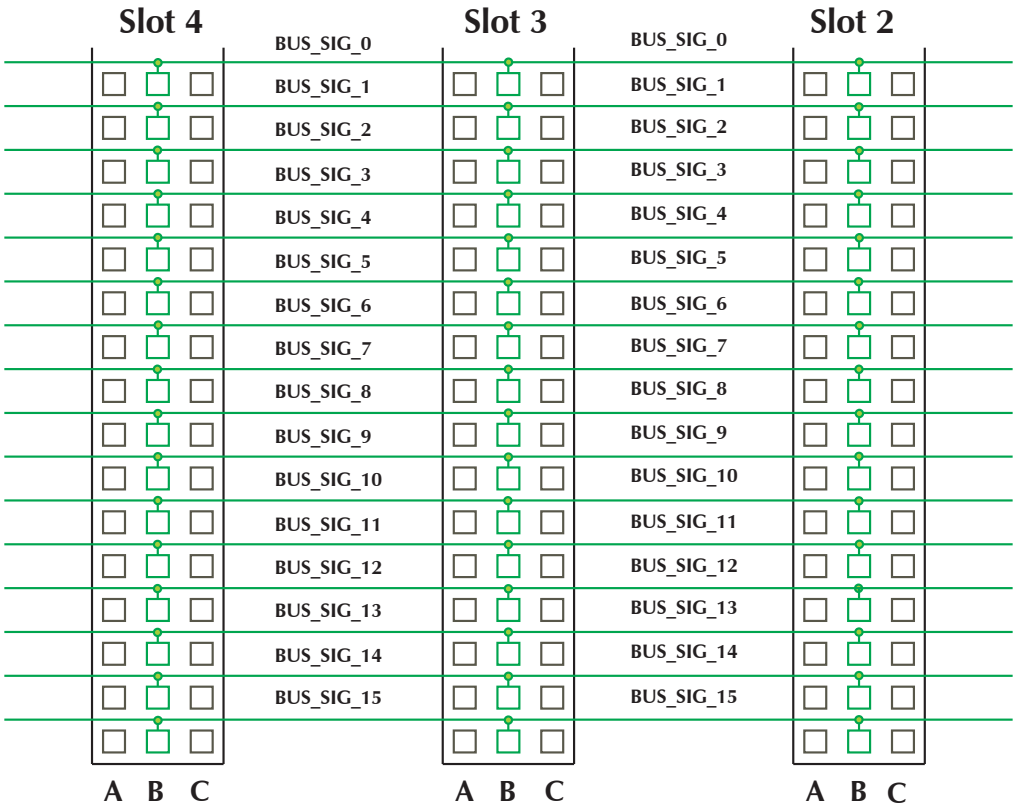


Analog Backplane

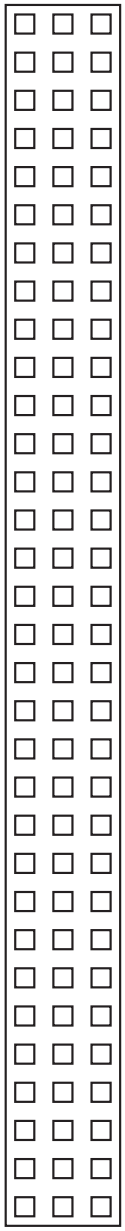
The analog backplane has two main purposes. One is to provide the analog power to RIC cards. Notice that the upper pins on the backplane, A1 - A16, B1 - B16, and C1 - C16 are used for power distribution for the RIC Power Supplies, and also signal lines for address decoding for the RIC cards.

The second purpose is to allow RIC cards to share signals. Signals can be shared in two ways.

The first is by using the 16 System Bus Signals provided by pins B17 - B32 on each RIC card. These 16 signals run the length of the Analog Backplane. They connect to the same pin on each of the slots and can be used to connect a signal from a RIC card to another RIC card, provided the cards have access to one or more of the 16 signals. The green signals on the bus below represent the System Bus Signals



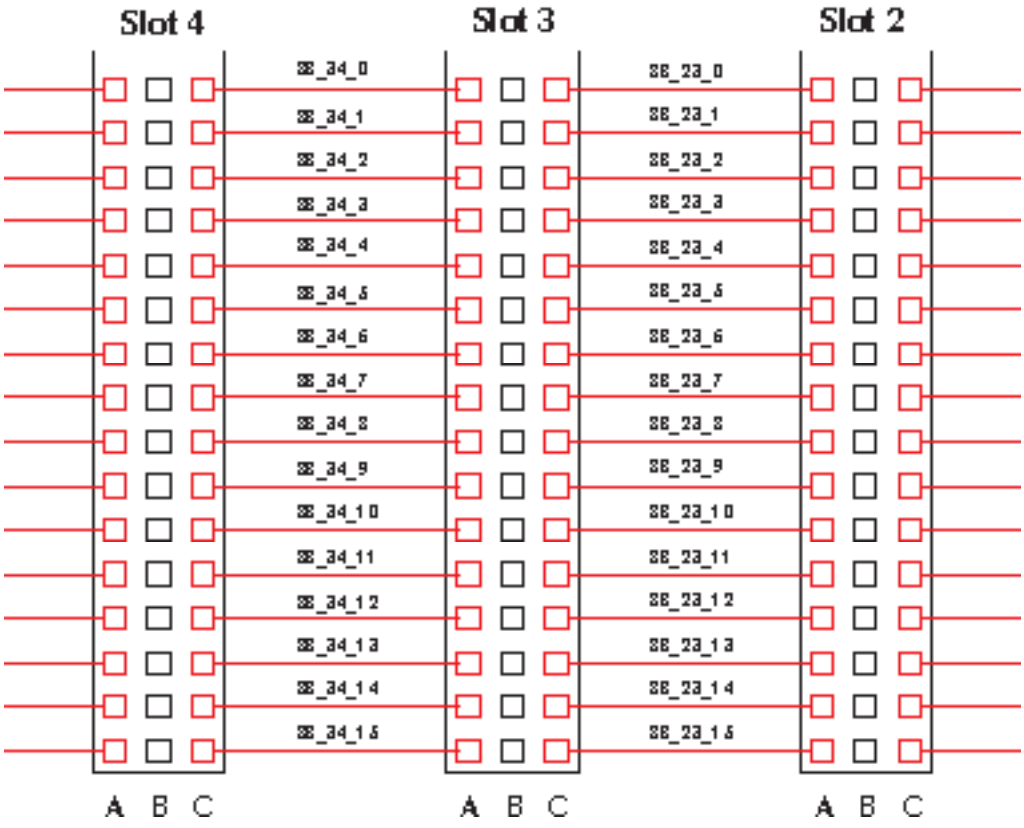
Typical Analog Backplane Slot (Slot #3)



A	B	C
A1 - +PS1	B1 - +PS1	C1 - +PS1
A2 - +PS0	B2 - +PS0	C2 - +PS0
A3 - +15V	B3 - +15V	C3 - +15V
A4 - +19.5V	B4 - +19.5V	C4 - +19.5V
A5 - +PS3	B5 - AGnd	C5 - +PS2
A6 - AGnd	B6 - AGnd	C6 - AGnd
A7 - AGnd	B7 - AGnd	C7 - AGnd
A8 - AGnd	B8 - AGnd	C8 - AGnd
A9 - AGnd	B9 - AGnd	C9 - AGnd
A10 - -PS3	B10 - AGnd	C10 - -PS2
A11 - -19.5V	B11 - -19.5V	C11 - -19.5V
A12 - -15V	B12 - -15V	C12 - -15V
A13 - -PS0	B13 - -PS0	C13 - -PS0
A14 - -PS1	B14 - -PS1	C14 - -PS1
A15 - N/C	B15 - N/C	C15 - /SEL_SLOT_3
A16 - /SEL_SLOT_3	B16 - /SEL_SLOT_3	C16 - /SEL_SLOT_3
A17 - SB_23_0	B17 - BUS_SIG_0	C17 - SB_34_0
A18 - SB_23_1	B18 - BUS_SIG_1	C18 - SB_34_1
A19 - SB_23_2	B19 - BUS_SIG_2	C19 - SB_34_2
A20 - SB_23_3	B20 - BUS_SIG_3	C20 - SB_34_3
A21 - SB_23_4	B21 - BUS_SIG_4	C21 - SB_34_4
A22 - SB_23_5	B22 - BUS_SIG_5	C22 - SB_34_5
A23 - SB_23_6	B23 - BUS_SIG_6	C23 - SB_34_6
A24 - SB_23_7	B24 - BUS_SIG_7	C24 - SB_34_7
A25 - SB_23_8	B25 - BUS_SIG_8	C25 - SB_34_8
A26 - SB_23_9	B26 - BUS_SIG_9	C26 - SB_34_9
A27 - SB_23_10	B27 - BUS_SIG_10	C27 - SB_34_10
A28 - SB_23_11	B28 - BUS_SIG_11	C28 - SB_34_11
A29 - SB_23_12	B29 - BUS_SIG_12	C29 - SB_34_12
A30 - SB_23_13	B30 - BUS_SIG_13	C30 - SB_34_13
A31 - SB_23_14	B31 - BUS_SIG_14	C31 - SB_34_14
A32 - SB_23_15	B32 - BUS_SIG_15	C32 - SB_34_15

The second method to share signals between cards is to use the 16 Bus Left and 16 Bus Right Signals as shown by the red signals below. These signals create a daisy-chained bus that connects each RIC card with its adjacent card to the left and to the right. Thus, the Bus Right Signal of a given RIC slot connects to the Bus Left signal of the adjacent slot, and so on. Slot one only has the 16 Bus Right Signals and slot 18 only has the 16 Bus Left Signals.

The System Bus Signals and the Bus Left/Right Signals can handle 100 volts and 0.750 amps.



Digital Backplane

The digital backplane has three main purposes. One is to provide the digital power to the RIC cards. The second is to provide the parallel or serial communication from the 2050 PXI Bus Controller card and the RIC cards. The third is to provide the 68 Pass Through Signals from the PXI rack to the RIC cards.

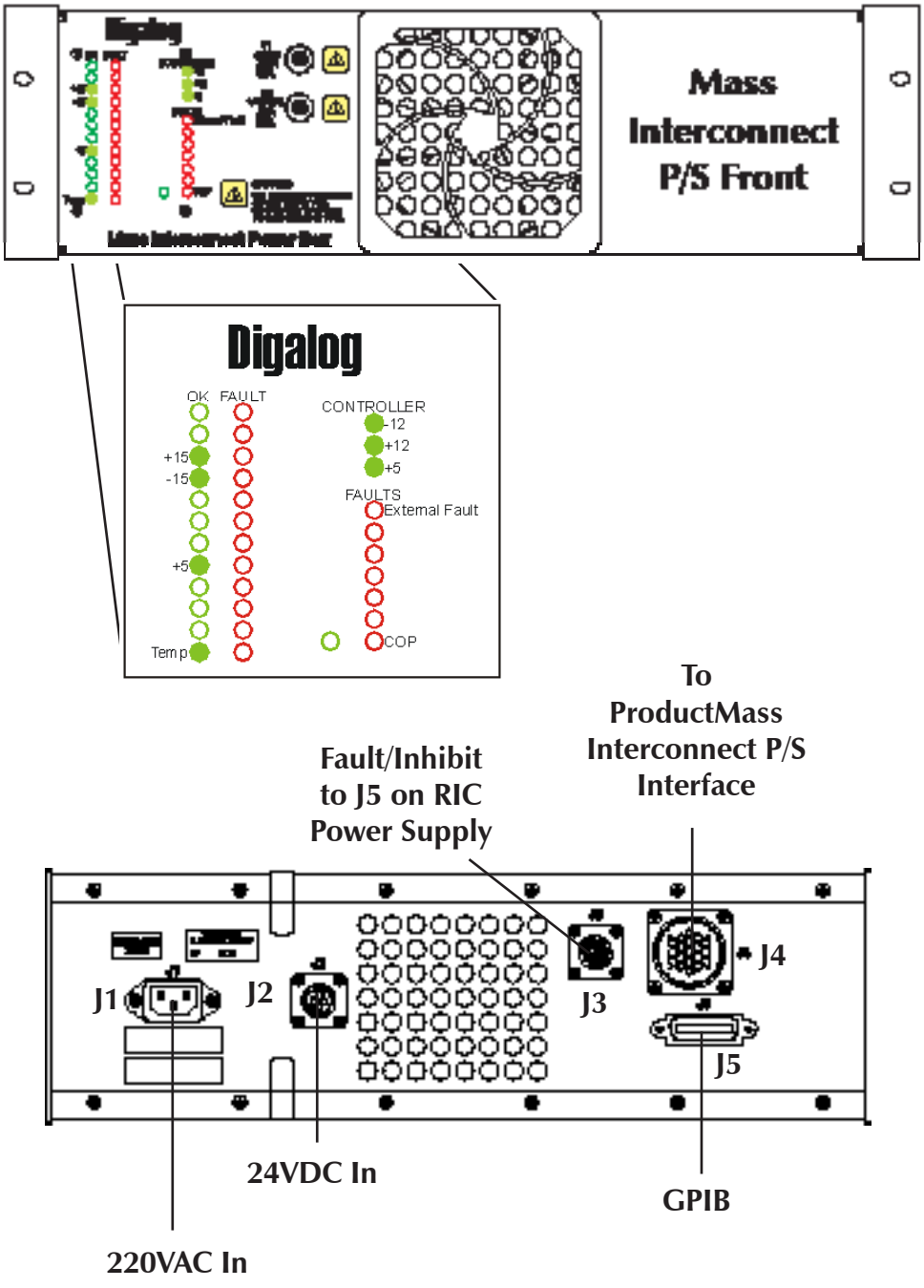
The main use for the Pass Through Signals is to provide a means to get the inputs and outputs from a 3U PXI instrument card up to the RIC cards or out to the Mass Interconnect Panel. This is accomplished by installing a 3U Pass Through card above the 3U PXI instrument card. The PXI instrument card's inputs and outputs can be connected to the Pass Through card with external cables. The Pass Through cards will then route the 68 signals to the digital backplane which then routes the signals to the RIC slot. Typically the RIC card will then route the Pass Through Signals to the Mass Interconnect Panel. In addition, some RIC cards will provide a means to connect their circuitry to the Pass Through Signals. An example is the 128x4 Matrix card that allows the user to connect eight of the Pass Through Signals to the Matrix card's buses.

Please note, the 68 Pass Through Signals are shared with the parallel communication signals. Therefore, the Pass Through Signals are only available in RIC slots that do not have a parallel communication card in them.

The 68 Pass Through signals can handle 100 volts and 0.375 amps.

Mass Interconnect P/S

Mass Interconnect P/S



Mass Interconnect Power Supply Assembly

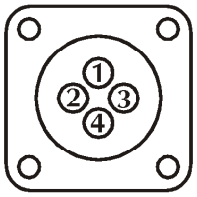
The Mass Interconnect power supplies offer utility voltages of +5V@3A and ±15V@0.8A for use in powering either the UUT or circuitry located in the fixture. These power supplies are programmed (on or off) using a software functional call. The outputs of the supplies are also monitored for fault conditions. A fault on the power supplies will only affect the UUT power supply system (the RIC power supplies are unaffected). If a fault is detected, the Mass Interconnect power supplies are disconnected from both the AC power and the Mass Interconnect Panel. The supplies are also disconnected from the Mass Interconnect Panel when they are programmed to an off state.

The assembly contains a GPIB controller used for monitoring the sense outputs from the power supplies and fault loop control. The front panel also contains a series of LED output indicators for monitoring the status of this assembly and the RIC Power Supply assembly as shown to the left. Fuse assignments are shown below.

F1	**±15VDC**	**0.375A**
F2	**+5VDC**	**0.25A**
F3	**24VDC In**	**1A**

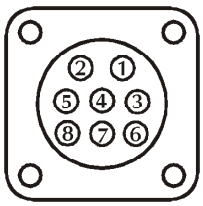
J2 - 24VDC In

Connector	Signal
1	24VDC
2	N/C
3	GND
4	N/C



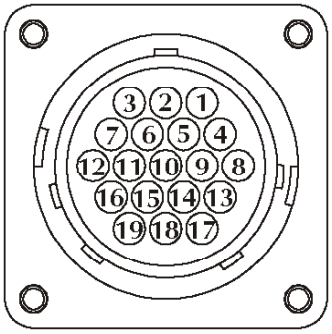
J3 - Fault/Inhibit

Connector	Signal
1	FAULT_LOOP_OUT
2	GND
3	N/C
4	N/C
5	N/C
6	N/C
7	PS1_FAULT_IN_COL
8	PS1_FAULT_IN_EMTR

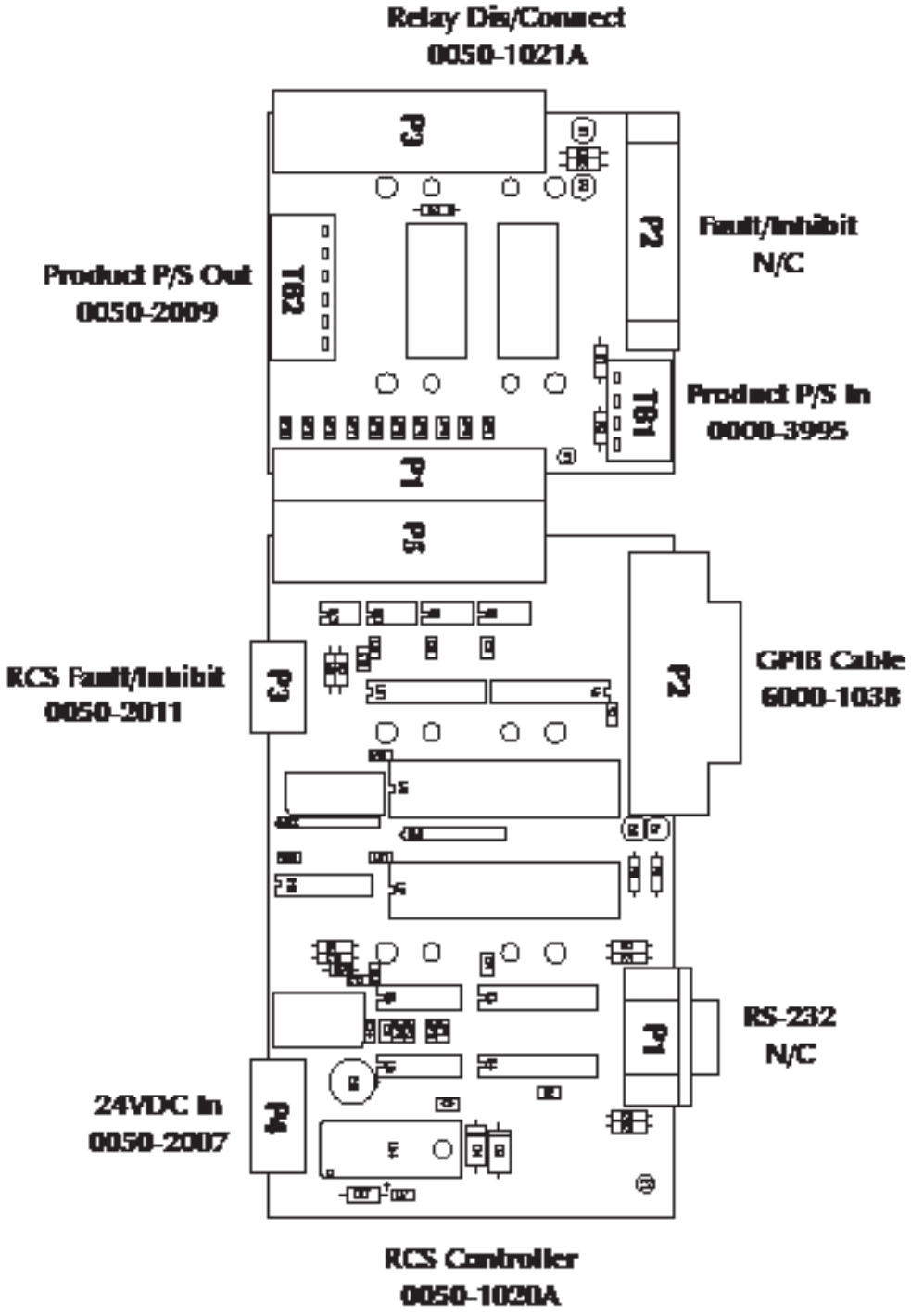


J4 - Patchboard Power

Connector	Signal
1	N/C
2	Keying Plug
3	N/C
4	+5VDC
5	+5V Sense +
6	+5V Common
7	+5V Sense -
8	N/C
9	+15V Sense +
10	+15VDC
11	+15V Common
12	+15V Sense -
13	-15V Sense - (- sense)
14	-15VDC
15	-15V Common (+out)
16	-15V Sense + (+ sense)
17	N/C
18	N/C
19	N/C



24VDC RCS Controller Assembly


Relay Dis/Connect
0050-1021A
P3
P2
Fault/Inhibit
N/C
Product P/S Out
0050-2009
TB2
TB1
Product P/S In
0000-3995
P1
P5
RCS Fault/Inhibit
0050-2011
P6
P2
GPIB Cable
6000-1038
P1
RS-232
N/C
24VDC In
0050-2007
P4
RCS Controller
0050-1020A

Relay Controller - 0050-1020A

The Relay Controller is designed to control the output relays of the GPIB power supplies on Digalog's Series 2050 Test System. It consists of one controller board plus one relay dis/connect board for each Product P/S output. A graphic of the system is shown on the left page.

The RCS is a complete system which provides relay control for up to five sets (Vout+, Vout-) of outputs from the product power supplies. The relays, which connect the outputs and the sense lines of the GPIB power supplies to the RIC, are managed by an on-board microcontroller which communicates with the host system controller through its IEEE-488 interface to effect the execution of each GPIB-related power functional call. The microcontroller also constantly monitors the system fault loop and disengages all of the relays (and thus the product power supplies' outputs) if a fault signal has been generated either by a functional call or by any of the product power supplies. This prevents the product power supplies from possibly damaging or being damaged by the product. It also has a selftest mode to check its own functionality.

TB1 - Relay Dis/Connect - Product P/S In
1 - +Out - 14AWG White
2 - +Out - 14AWG White
3 - +Sense - 20AWG White/Red
4 - -Sense - 20AWG White/Black
5 - -Out - 14AWG Black
6 - -Out - 14AWG Black

TB2 - Relay Dis/Connect - Product P/S to RIC - 0050-2013
1 - +Out - 14AWG White
2 - +Out - 14AWG White
3 - +Sense - 20AWG Black/White
4 - -Sense - 20AWG Black
5 - -Out - 14AWG Black
6 - -Out - 14AWG Black

P2 - Relay Dis/Connect - Fault/Inhibit From PPS
1 - +Fault
2 - -Fault
3 - -Inhibit
4 - -Inhibit
5 - N/C

6 - N/C
7 - Shield
8 - N/C

P3 - Relay Controller - Fault/Inhibit - #0050-2011
1 - +Fault
2 - -Fault
3 - +Inhibit
4 - -Inhibit

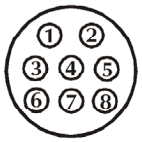
P4 - Relay Controller - 24VDC In - #0050-2007
1 - Shield
2 - N/C
3 - Gnd
4 - N/C
5 - 24VDC

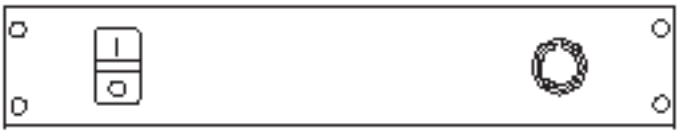
E-Stop/Power On/Off Panel

Remote E-Stop/Power Switch Assembly

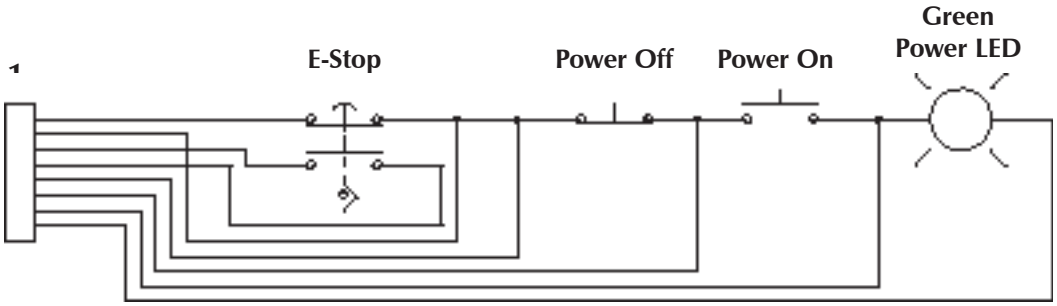
The back panel of the AC Power Management assembly contains a connector labeled, J14, “ESTOP ON/OFF”, that is provided to connect with the remote power control panel. The remote power control panel contains the E-Stop switch and the main power On/Off switch mounted to a 2U panel.

Pin#	Signal
1	ESTOP NC-1
2	ESTOP NC-2
3	ESTOP NO-1
4	ESTOP NO-2
5	POWER OFF
6	POWER ON/OFF
7	POWER ON
8	POWER LIGHT RETURN





0050-5101 E-Stop/Power Switch Assembly

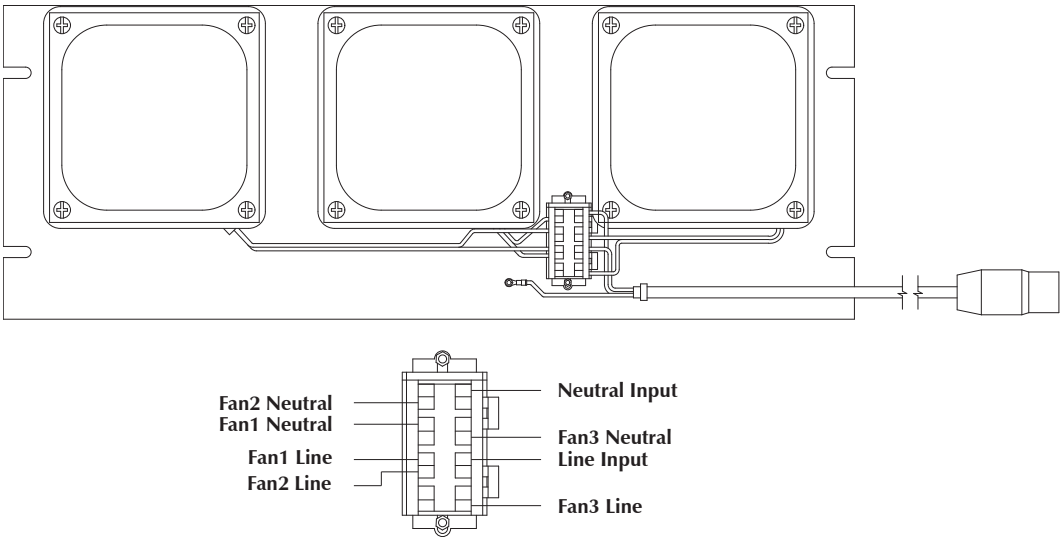


AC Fan Assembly

AC Fan Assembly

The lower rear of the cabinet assembly contains a set of three fans for circulation and cooling. This assembly, #0003-0006, is powered by 220VAC as shown below.

AC Fan Assembly #0003-0006

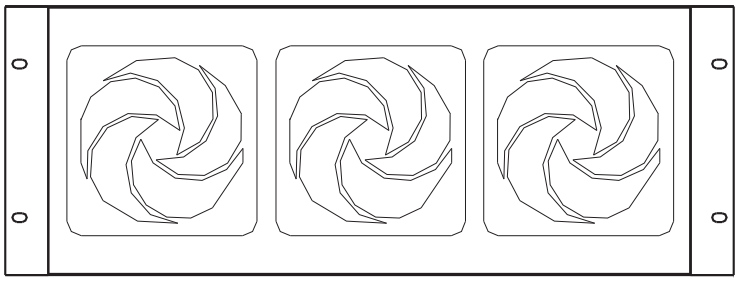


PXI Chassis and Mass Interconnect Fans

PXI Chassis and Mass Interconnect Panel Fan Assemblies

Both the Mass Interconnect Panel and the PXI chassis are cooled by DC fan assemblies. These assemblies contain three DC fans powered by 24VDC from the AC Power Management assembly. Both fan assemblies are identical except for mounting hardware. The PXI chassis fan assembly is #0050-0033 and the Mass Interconnect Panel fan assembly is #0050-0035.

Fan Assemblies #0050-0033/0035

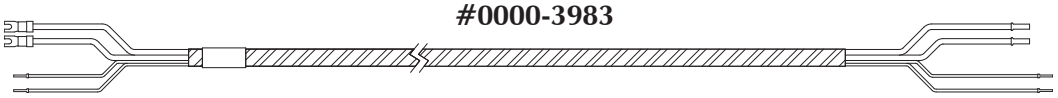


System Cables

System Cables

0000-3983	Product P/S to RCS Relay Dis/Connect
0000-5928	208 - 240VAC In
0000-5934	208 - 240VAC AC Power Management to UPS
0050-2001	RIC P/S J3 to Analog Motherboard
0050-2002	RIC P/S J4 to Digital Motherboard
0050-2003	UPS REPO Cable
0050-2005	PXI Bus Power
0050-2006	24VDC From AC Power Management to Mass Interconnect Panel Fan Assembly, PXI Fan Assembly, Mass Interconnect P/S, RIC P/S
0050-2007	24VDC To RCS Controller
0050-2010	Fault Inhibit From Mass Interconnect P/S to J5 on RIC P/S
0050-2011	Fault Inhibit from RCS Controller To J6 on RIC P/S
0050-2012	Mass Interconnect P/S To RIC
0050-2013	Relay Dis/Connect To RIC
0050-2014	Fault/Inhibit from Proximity Switch to J7 on RIC P/S
6000-1021	220VAC Power Cable (6)
6000-1038	GPIB to RCS
6000-1039	GPIB (3)

Agilent 6642A Output to Relay Dis/Connect
#0000-3983

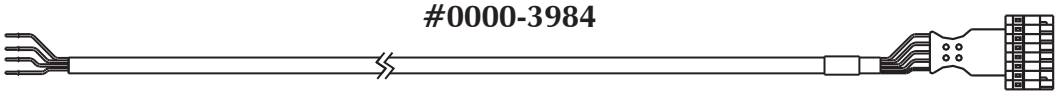


Wht/Blk 20AWG - Sense-
Wht/Red 20AWG - Sense+

TB1
+Out - 14AWG White - 1 or 2
+Sense - 20AWG White/Red - 3
-Sense - 20AWG White/Black - 4
-Out - 14AWG Black - 5 or 6

N/C
Black 14AWG - Out-
White 14AWG - Out+

Agilent 6642A Fault/Inhibit to P2 on Relay Dis/Connect
#0000-3984



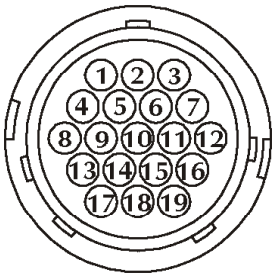
+Fault - White
-Fault - Green
-Inhibit - Black
+Inhibit - Red

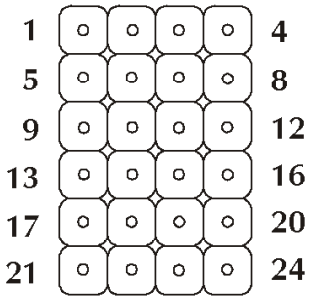
P2 on #0050-1021 Relay Dis/Connect

+Fault - White - 1
-Fault - Green - 2
-Inhibit - Black - 3
+Inhibit - Red - 4
N/C - 5
N/C - 6
Shield - 7
N/C - 8

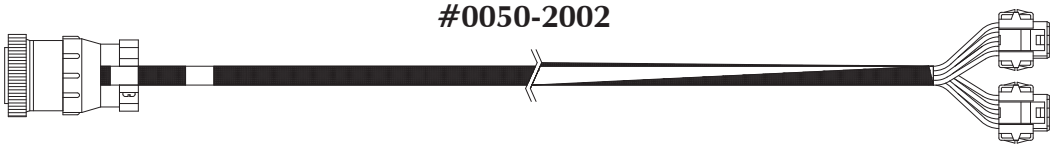
RIC P/S P3 To J19 On Analog Motherboard
#0050-2001

1 - +15V Sense+	+15V Sense+ - 1
2 - +15VDC	+15VDC - 2
3 - +15V Sense-	+15V Sense- - 3
4 - +15V Common	+15V Common - 4
5 - +19V Sense+	+19V Sense+ - 5
6 - +19VDC	+19VDC - 6
7 - +19V Sense-	+19V Sense- - 7
8 - +19V Common	+19V Common - 8
9 - -19V Sense- (-Sense)	N/C - 9
10 - -19VDC	N/C - 10
11 - -19V Sense+ (+Sense)	N/C - 11
12 - -19V Common (+Out)	N/C - 12
13 - -15V Sense- (-Sense)	N/C - 13
14 - -15VDC	N/C - 14
15 - -15V Sense+(+Sense)	N/C - 15
16 - -15V Common (+Out)	N/C - 16
17 - N/C	-19V Sense- (-Sense) - 17
18 - N/C	-19VDC - 18
19 - N/C	-19V Sense+ (+Sense) - 19
	-19V Common (+Out) - 20
	-15V Sense- (-Sense) - 21
	-15VDC - 22
	-15V Sense+(+Sense) - 23
	-15V Common (+Out) - 24

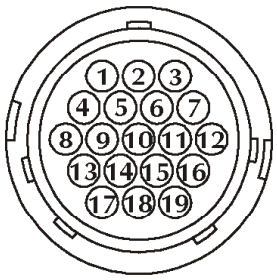




RIC P/S J4 To J37 & J38 On Digital Motherboard
#0050-2002

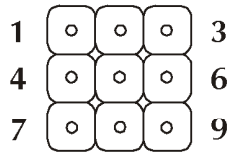


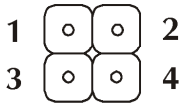
1 - +5VDC
2 - +5VDC
3 - +5V Sense+
4 - N/C
5 - +5V Common
6 - +5V Common
7 - +5V Sense-
8 - N/C
9 - N/C
10 - EXT_PS_FAULT_LOOP_OUT
11 - PS1_FAULT_COL
12 - PS1_FAULT_EMTR
13 - GND
14 - GND
15 - N/C
16 - +48V Sense+
17 - +48VDC
18 - +48V Sense-
19 - +48V Common



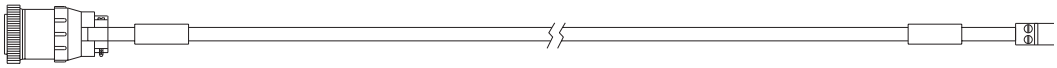
J38
+5VDC - 1
+5VDC - 2
+5V Sense+ - 3
PS1_FAULT_COL - 4
EXT_PS_FAULT_LOOP_OUT - 5
PS1_FAULT_EMTR/GND - 6
+5V Sense- - 7
+5V Common - 8
+5V Common - 9

J37
+48V Sense+ - 1
+48VDC - 2
+48V Common - 3
+48V Sense- - 4

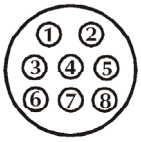




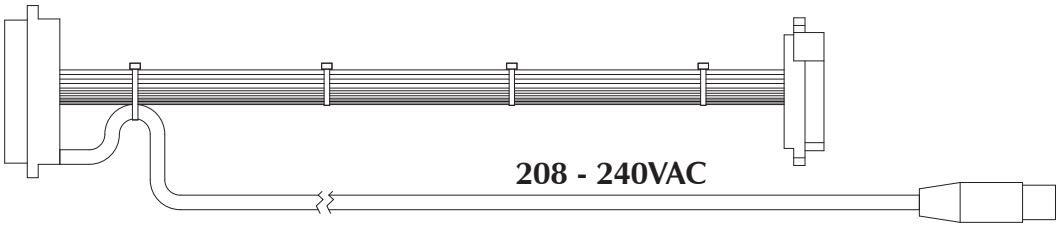
AC Power Management to UPS REPO Cable
#0050-2003



J15
1 - N/C
2 - N/C
3 - To REPO P1
4 - To REPO P2
5 - N/C
6 - N/C
7 - N/C
8 - N/C

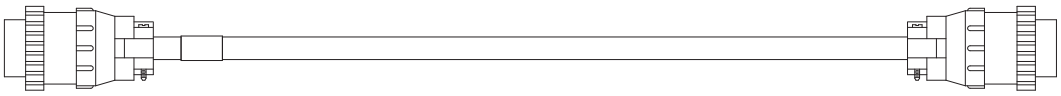


PXI Bus Power Cable
#0050-2005

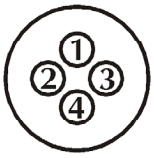


P1 To Power One Supply		P2 Tracewell Backplane
1 - +5VDC	25 - N/C	+5VDC - 1
2 - +5VDC	26 - Reserved	+5VDC - 2
3 - +5VDC	27 - Enable	+5/+3.3 Rtn - 3
4 - +5VDC	28 - N/C	+5/+3.3 Rtn - 4
5 - +5/+3.3 Rtn	29 - N/C	+12VDC Rtn - 5
6 - +5/+3.3 Rtn	30 - +5V Remote Sense	+12VDC - 6
7 - +5/+3.3 Rtn	31 - N/C	+12V Remote Sense - 7
8 - +5/+3.3 Rtn	32 - N/C	+12VDC Rtn - 8
9 - +5/+3.3 Rtn	33 - +3.3V Remote Sense	+12VDC - 9
10 - +5/+3.3 Rtn	34 - Sense Rtn	Enable - 10
11 - +5/+3.3 Rtn	35 - +5V Current Share	-12VDC Rtn - 11
12 - +5/+3.3 Rtn	36 - +12V Remote Sense	-12VDC - 12
13 - +3.3VDC	37 - N/C	Inhibit - 13
14 - +3.3VDC	38 - Degrade Signal	N/C - 14
15 - +3.3VDC	39 - Inhibit	N/C - 15
16 - +3.3VDC	40 - N/C	Signal Rtn - 16
17 - +3.3VDC	41 - +3.3V Current Share	+5V Remote Sense - 17
18 - +3.3VDC	42 - Fail Signal	Sense Rtn - 18
19 - +12VDC Rtn	43 - N/C	N/C - 19
20 - +12VDC	44 - +12V Current Share	+3.3V Remote Sense - 20
21 - -12VDC	45 - Chassis Gnd	N/C - 21
22 - Signal Rtn	46 - AC Input Neutral	N/C - 22
23 - Reserved	47 - AC Input Line	N/C - 23
24 - -12VDC Rtn		N/C - 24
		N/C - 25
		+5/+3.3 Rtn - 26
		+5/+3.3 Rtn - 27
		+3.3VDC - 28
		+3.3VDC - 29

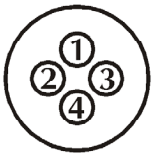
AC Power Management (24VDC) to Mass Interconnect Fan Assembly, PXI Fan Assembly, Mass Interconnect P/S, & RIC P/S #0050-2006



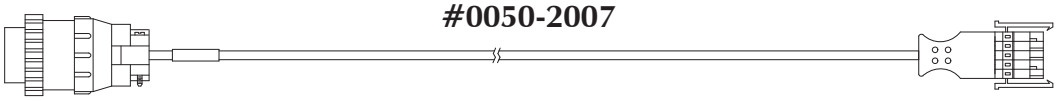
1 - 24VDC
2 - N/C
3 - Ground
4 - N/C



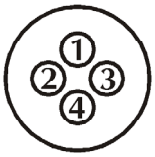
24VDC - 1
N/C - 2
Ground - 3
N/C - 4



AC Power Management To RCS Controller
#0050-2007

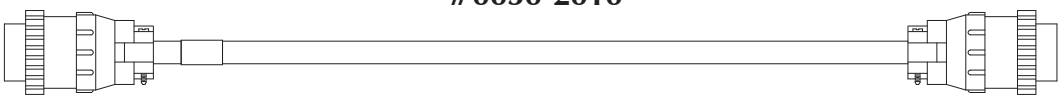


1 - 24VDC
2 - N/C
3 - Ground
4 - N/C

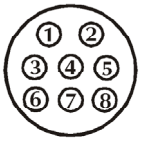


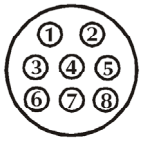
Shield - 1
N/C - 2
Ground - 3
N/C - 4
24VDC - 5

Fault Inhibit From Mass Interconnect P/S to J5 on Testhead P/S #0050-2010

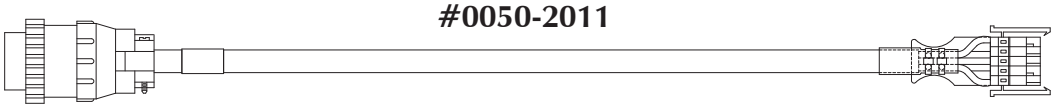


1 - Fault Loop Out	PS1_Fault_In_Col - 1
2 - Gnd	PS1_Fault_In_Emtr - 2
3 - N/C	N/C - 3
4 - N/C	N/C - 4
5 - N/C	N/C - 5
6 - N/C	N/C - 6
7 - PS1_Fault_In_Col	Fault Loop Out - 7
8 - PS1_Fault_In_Emtr	Gnd - 8





RIC P/S J6 To Fault/Inhibit On RCS Controller
#0050-2011

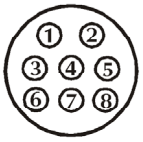


P1

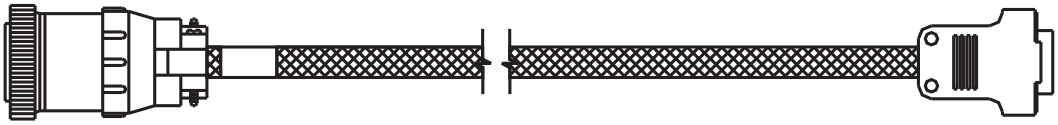
1 - Ext_Fault_2
2 - Gnd
3 -
4 -
5 - Fault_Out2_Col
6 - Fault_Out2_Emtr
7 - N/C
8 - N/C

P2 - RCS Controller

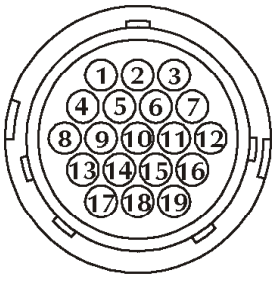
Ext_Fault_2 - 1
Gnd - 2
Fault_Out2_Col - 3
Fault_Out2_Emtr - 4



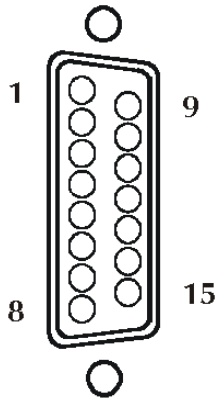
Mass Interconnect P/S To RIC
#0050-2012



Patchboard P/S
1 - N/C
2 - Keying Plug
3 - N/C
4 - +5VDC
5 - +5V Sense+
6 - +5V Common
7 - +5V Sense-
8 - N/C
9 - +15V Sense+
10 - +15VDC
11 - +15V Common
12 - +15V Sense-
13 - -15V Sense-
14 - -15VDC
15 - -15V Common
16 - -15V Sense+
17 - N/C
18 - N/C
19 - N/C



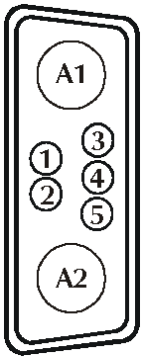
Testhead
+5VDC - 1
+5V Sense- - 2
N/C - 3
+15V Sense+ - 4
+15V Sense- - 5
N/C - 6
-15V Sense+ - 7
-15V Sense- - 8
+5V Sense+ - 9
+5V Common - 10
+15VDC - 11
+15V Common - 12
N/C - 13
-15V Common - 14
-15VDC - 15



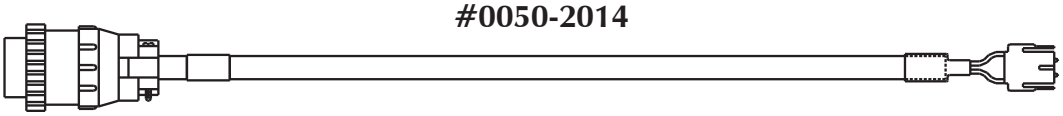
PPS0 UUT Power From Relay Dis/Connect
#0500-2013

A1 - +Out1, +Out2
A2 - -Out1, -Out2
1 - N/C
2 - N/C
3 - +Sense
4 - N/C
5 - -Sense

TB2
+Out1 - 14AWG White - 1
+Out2 - 14AWG White - 2
+Sense - 20AWG Blk/Wht - 3
-Sense - 20AWG Black- 4
-Out1 - 14AWG Black - 5
-Out2 14AWG Black - 6

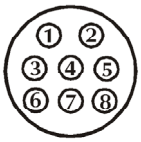


RIC J7 To Prox Switch Fault/Inhibit
#0050-2014



P1

1 - Ext_Fault_3
2 - Gnd
3 - Blank Pin
4 - N/C
5 - N/C
6 - N/C
7 - N/C
8 - N/C



2050 Switch Family

Introduction:

This instrument driver provides programming support for the 2050 RIC Switch Family. It contains functions for opening, configuring, taking measurements from, and closing the instrument.

Assumptions:

To successfully use this module, the following conditions must be met:

For GPIB instrument drivers:

- the instrument is connected to the GPIB.
- the GPIB address supplied to the initialize function must match the GPIB address of the instrument.

For VXI instrument drivers:

- the instrument is installed in the VXI mainframe and you are using one of the following controller options:
  - Embedded controller
  - MXI
  - MXI2
  - GPIB-VXI
- the logical address supplied to the initialize function must match the logical address of the instrument.

For RS-232 instrument drivers:

- the instrument is connected to the RS-232 interface.
- the COM port, baud rate, parity, and timeout supplied to the initialize function must match the settings of the instrument.

Error and Status Information:

Each function in this instrument driver returns a status code that either indicates success or describes an error or warning condition. Your program should examine the status code from each call to an instrument driver function to determine if an error occurred. The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

Use this document as a programming reference manual. It describes each function in the 2050 RIC Switch Family instrument. The functions appear in alphabetical order, with a description of the function and its C syntax, a description of each parameter, and a list of possible error codes.

Function Tree Layout:

Class/Panel Name:	Function Name:
Initialize	dl50Sw_init
Initialize With Options	dl50Sw_InitWithOptions
Configuration	
Set/Get/Check Attribute	
Set Attribute	
Set Attribute ViInt32	dl50Sw_SetAttributeViInt32
Set Attribute ViReal64	dl50Sw_SetAttributeViReal64
Set Attribute ViString	dl50Sw_SetAttributeViString
Set Attribute ViBoolean	dl50Sw_SetAttributeViBoolean
Set Attribute ViSession	dl50Sw_SetAttributeViSession
Get Attribute	
Get Attribute ViInt32	dl50Sw_GetAttributeViInt32
Get Attribute ViReal64	dl50Sw_GetAttributeViReal64
Get Attribute ViString	dl50Sw_GetAttributeViString
Get Attribute ViBoolean	dl50Sw_GetAttributeViBoolean
Get Attribute ViSession	dl50Sw_GetAttributeViSession
Check Attribute	
Check Attribute ViInt32	dl50Sw_CheckAttributeViInt32
Check Attribute ViReal64	dl50Sw_CheckAttributeViReal64
Check Attribute ViString	dl50Sw_CheckAttributeViString
Check Attribute ViBoolean	dl50Sw_CheckAttributeViBoolean
Check Attribute ViSession	dl50Sw_CheckAttributeViSession
Route	
Connect Channels	dl50Sw_Connect
Disconnect Channels	dl50Sw_Disconnect
Disconnect All Channels	dl50Sw_DisconnectAll
Switch Is Debounced?	dl50Sw_IsDebounced
Wait For Debounce	dl50Sw_WaitForDebounce
Can Connect Channels?	dl50Sw_CanConnect

Paths	
Set Path	dl50Sw_SetPath
Get Path	dl50Sw_GetPath
Utility	
Reset	dl50Sw_reset
Reset With Defaults	dl50Sw_ResetWithDefaults
Disable	dl50Sw_Disable
Self-Test	dl50Sw_self_test
Revision Query	dl50Sw_revision_query
Error-Query	dl50Sw_error_query
Error Message	dl50Sw_error_message
Invalidate All Attributes	dl50Sw_InvalidateAllAttributes
Get Channel Name	dl50Sw_GetChannelName
Error Info	
Get Error	dl50Sw_GetError
Clear Error	dl50Sw_ClearError
Coercion Info	
Get Next Coercion Record	dl50Sw_GetNextCoercionRecord
Interchangeability Info	
Get Next Interchange Warning	dl50Sw_GetNextInterchangeWarning
Clear Interchange Warnings	dl50Sw_ClearInterchangeWarnings
Reset Interchange Check	dl50Sw_ResetInterchangeCheck
Locking	
Lock Session	dl50Sw_LockSession
Unlock Session	dl50Sw_UnlockSession
Close	dl50Sw_close

This instrument driver contains programming support for the following cards:

1510 Digalog Systems 2050 Auxiliary Relay Card.
1536 Digalog Systems 2050 Matrix Relay Card.
1574 Digalog Systems 2050 Bus Controller Card

This driver has all the functions that IVI requires.

Note: This driver requires NI-VISA and NI IVI libraries.

dl50Sw_CanConnect

This function verifies that the switch module is capable of creating a path between the two channels you specify with the Channel 1 and Channel 2 parameters. If the switch module is capable of creating a path, this function indicates whether the path is currently available given the existing connections.

If the path is not available due to the currently existing connections, but the implicit connection between the two channels already exists, the function returns the DL50SW_WARN_IMPLICIT_CONNECTION_EXISTS (0x3FFA2002) warning.

PROTOTYPE: ViStatus dl50Sw_CanConnect (ViSession instrumentHandle, ViChar _VI_FAR channel1[], ViChar _VI_FAR channel2[], ViPInt32 pathCapability);

<int> = dl50Sw_CanConnect(ViSession instrumentHandle, channel1[], channel2[], pathCompatibility);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channel1 You identify a path with two channels. Pass one of the channel names for which you want to create a path. Pass the other channel name as the Channel 2 parameter.

channel2 You identify a path with two channels. Pass one of the channel names for which you want to create a path. Pass the other channel name as the Channel 1 parameter.

pathCapability Indicates whether a path is valid. Possible values include:

Status Name	Actual Value
DL50SW_VAL_PATH_AVAILABLE	1
DL50SW_VAL_PATH_EXISTS	2
DL50SW_VAL_PATH_UNSUPPORTED	3
DL50SW_VAL_RSRC_IN_USE	4
DL50SW_VAL_SOURCE_CONFLICT	5
DL50SW_VAL_CHANNEL_NOT_AVAILABLE	6

Notes:

(1) DL50SW_VAL_PATH_AVAILABLE indicates that the driver can create the path at this time.

(2) DL50SW_VAL_PATH_EXISTS indicates that the path already exists.

(3) DL50SW_VAL_PATH_UNSUPPORTED indicates that the instrument is not capable of creating a path between the channels you specify.

(4) DL50SW_VAL_RSRC_IN_USE indicates that although the path is valid, the driver cannot create the path at this moment because the switch module is currently using one or more of the required channels to create another path. You must delete the other path before creating this one.

(5) DL50SW_VAL_SOURCE_CONFLICT indicates that the instrument cannot create a path because both channels are connected to a different source channel.

(6) DL50SW_VAL_CHANNEL_NOT_AVAILABLE indicates that the driver cannot create a path between the two channels because one of the channels is a configuration channel and thus unavailable for external connections.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_CheckAttributeViBoolean

This function checks the validity of a value you specify for a ViBoolean attribute.

PROTOTYPE: ViStatus dl50Sw_CheckAttributeViBoolean (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViBoolean attributeValue);

<int> = dl50Sw_CheckAttributeViBoolean(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViBoolean type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViBoolean are dim. If you select an attribute data type that is dim, LabWindows/

CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_CheckAttributeViInt32

This function checks the validity of a value you specify for a ViInt32 attribute.

PROTOTYPE: ViStatus dl50Sw_CheckAttributeViInt32 (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViInt32 attributeValue);

<int> = dl50SwCheckAttributeViInt32(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViInt32 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_CheckAttributeViReal64

This function checks the validity of a value you specify for a ViReal64 attribute.

PROTOTYPE: ViStatus dl50Sw_CheckAttributeViReal64 (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViReal64 attributeValue);

<int> = dl50Sw_CheckAttributeViReal64(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press <ENTER>, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViReal64 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViReal64 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function**.**

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_CheckAttributeViSession

This function checks the validity of a value you specify for a ViSession attribute.

PROTOTYPE: ViStatus dl50Sw_CheckAttributeViSession (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViSession attributeValue);

<int> = dl50Sw_CheckAttributeViSession(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViSession type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViSession are dim. If you select an attribute data type that is dim, LabWindows/

CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function**.**

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_CheckAttributeViString

This function checks the validity of a value you specify for a ViString attribute.

PROTOTYPE: ViStatus dl50Sw_CheckAttributeViString (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViChar _VI_FAR attributeValue[]);

<int> = dl50Sw_CheckAttributeViString(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViString type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViString are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function**.**

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_ClearError

This function clears the error code and error description for the IVI session. If the user specifies a valid IVI session for the InstrumentHandle parameter, this function clears the error information for the session. If the user passes VI_NULL for the Vi parameter, this function clears the error information for the current execution thread. If the Vi parameter is an invalid session, the function does nothing and returns an error.

The function clears the error code by setting it to VI_SUCCESS. If the error description string is non-NULL, the function de-allocates the error description string and sets the address to VI_NULL.

Maintaining the error information separately for each thread is useful if the user does not have a session handle to pass to the dl50Sw_GetError function, which occurs when a call to dl50Sw_init or dl50Sw_InitWithOptions fails.

PROTOTYPE: ViStatus dl50Sw_ClearError (ViSession instrumentHandle);

<int> = dl50Sw_ClearError(ViSession instrumentHandle);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_ClearInterchangeWarnings

This function clears the list of current interchange warnings.

PROTOTYPE: ViStatus dl50Sw_ClearInterchangeWarnings (ViSession instrumentHandle);

<int> = dl50Sw_ClearInterchangeWarnings(ViSession instrumentHandle);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings

3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_close

This function performs the following operations:

- Closes the instrument I/O session.
- Destroys the instrument driver session and all of its attributes.
- Deallocates any memory resources the driver uses.

Notes:

(1) You must unlock the session before calling dl50Sw_close.

(2) After calling dl50Sw_close, you cannot use the instrument driver again until you call dl50Sw_init or dl50Sw_InitWithOptions.

PROTOTYPE: ViStatus dl50Sw_close (ViSession instrumentHandle);

<int> = dl50Sw_close

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_Connect

This function creates a path between Channel 1 and Channel 2. The driver calculates the shortest path between the two channels. If a path is not available, the function returns one of the following errors:

DL50SW_ERROR_EXPLICIT_CONNECTION_EXISTS (0xBFFA200C), if the two channels are already explicitly connected by calling either the dl50Sw_Connect or dl50Sw_SetPath function.

DL50SW_ERROR_IS_CONFIGURATION_CHANNEL (0xBFFA2009), if a channel is a configuration channel. Error elaboration contains information about which of the two channels is a configuration channel.

DL50SW_ERROR_ATTEMPT_TO_CONNECT_SOURCES (0xBFFA200B), if both channels are connected to a different source. Error elaboration contains information about sources channel 1 and 2 connect to.

DL50SW_ERROR_CANNOT_CONNECT_TO_ITSELF (0xBFFA2015), if channels 1 and 2 are one and the same channel.

DL50SW_ERROR_PATH_NOT_FOUND (0xBFFA2011), if the driver cannot find a path between the two channels.

Notes:

(1) The paths are bidirectional. For example, if a path exists between channels CH1 and CH2, then the path between channels CH2 and CH1 also exists.

PROTOTYPE: dl50Sw_Connect (ViSession instrumentHandle, ViChar _VI_FAR channel1[], ViChar _VI_FAR channel2[]);

<int> = dl50Sw_Connect(ViSession instrumentHandle, channel1[], channel2[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channel1

You identify a path with two channels. Pass one of the channel names for which you want to create a path. Pass the other channel name as the Channel 2 parameter.

Valid Channel Names:

Model: 1510
A0, A1...A31
C0, C1...C31

Model: 1536
A0.....A7 (Analog Backplane)
PT48...PT55 (Pass Through)
I0.....I3 (Internal Pass Through and Analog bus)
BUS0...BUS3 (External channel bus)
CH0....CH127 (Channels)

channel2

You identify a path with two channels. Pass one of the channel names for which you want to create a path. Pass the other channel name as the Channel 1 parameter.

Valid Channel Names:

Model: 1510
A0, A1...A31
C0, C1...C31

Model: 1536
A0.....A7 (Analog Backplane)
PT48...PT55 (Pass Through)
I0.....I3 (Internal Pass Through and Analog bus)
BUS0...BUS3 (External channel bus)
CH0....CH127 (Channels)

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_Disable

This function places the instrument in a quiescent state where it has minimal or no impact on the system to which it is connected.

PROTOTYPE: ViStatus dl50Sw_Disable (ViSession instrumentHandle);

<int> = dl50Sw_Disable(ViSession instrumentHandle);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFF	VISA Warnings

3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_Disconnect

This function destroys the path between two channels that you create with the dl50Sw_Connect or dl50Sw_SetPath function.

PROTOTYPE: ViStatus dl50Sw_Disconnect (ViSession instrumentHandle, ViChar _VI_FAR channel1[], ViChar _VI_FAR channel2[]);

<int> = dl50Sw_Disconnect(ViSession instrumentHandle, channel1[], channel2[]);

instrumentHandle
The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channel1 You identify a path with two channels. Pass one of the channel names for which you want to create a path. Pass the other channel name as the Channel 2 parameter.

channel2 You identify a path with two channels. Pass one of the channel names for which you want to create a path. Pass the other channel name as the Channel 1 parameter.

Return Value Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function**.**

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_DisconnectAll

This function disconnects all existing paths.

Note: If the switch module is not capable of disconnecting all paths, this function returns DL50SW_WARN_PATH_REMAINS (0x3FFA2001) warning.

PROTOTYPE: ViStatus dl50Sw_DisconnectAll (ViSession instrumentHandle);

<int> = dl50Sw_DisconnectAll(ViSession instrumentHandle);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function**.**

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_error_message

This function converts a status code returned by an instrument driver function into a user-readable string.

PROTOTYPE: ViStatus dl50Sw_error_message (ViSession instrumentHandle, ViStatus errorCode, ViChar _VI_FAR errorMessage[]);

<int> = dl50Sw_error_message(ViSession instrumentHandle, errorCode, errorMessage[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

You can pass VI_NULL for this parameter. This is useful when one of the initialize functions fails.

Default Value: VI_NULL

errorCode

Pass the Status parameter that is returned from any of the instrument driver functions.

Default Value: 0 (VI_SUCCESS)

dl50Sw Status Codes:

Status	Description
WARNINGS:	
3FFA2001	Some connections remain after disconnecting.
3FFA2002	The channels are implicitly connected.
ERRORS:	
BFFA2001	Invalid path string.
BFFA2002	Invalid Scan List string.
BFFA2003	One of the channels is in use.
BFFA2004	The scan list string is empty.
BFFA2005	The path string is empty.
BFFA2006	The switch module is currently in scanning mode.
BFFA2007	The switch module is not currently in scanning mode.
BFFA2008	No explicit path exists between the two channels.
BFFA2009	Cannot explicitly connect a configuration channel.
BFFA200A	One path channel is not a configuration channel.
BFFA200B	Cannot connect two sources.

BFFA200C	The channels are explicitly connected.
BFFA200D	A leg in the path is missing the first channel name.
BFFA200E	A leg in the path is missing the second channel name.
BFFA200F	The first and second channels in the leg are the same.
BFFA2010	Duplicate channel in the path string.
BFFA2011	No path found between the channels.
BFFA2012	Beginning and end of adjacent legs are not the same.
BFFA2013	Path contains a leg with un-connectable channels.
BFFA2014	A leg in the path is already connected.
BFFA2015	A channel cannot be connected to itself.
BFFA4001	Hardware failed a write verification.

IviSwtch Status Codes:

Status	Description
WARNINGS:	
3FFA2001	Some connections remain after disconnecting.
3FFA2002	The channels are implicitly connected.
ERRORS:	
BFFA2001	Invalid path string.
BFFA2002	Invalid Scan List string.
BFFA2003	One of the channels is in use.
BFFA2004	The scan list string is empty.
BFFA2005	The path string is empty.
BFFA2006	The switch module is currently in scanning mode.
BFFA2007	The switch module is not currently in scanning mode.
BFFA2008	No explicit path exists between the two channels.
BFFA2009	Cannot explicitly connect a configuration channel.
BFFA200A	One path channel is not a configuration channel.
BFFA200B	Cannot connect two sources.
BFFA200C	The channels are explicitly connected.
BFFA200D	A leg in the path does not begin with a channel name.
BFFA200E	A leg in the path is missing the second channel name.
BFFA200F	The first and second channels in the leg are the same.
BFFA2010	Duplicate channel in the path string.
BFFA2011	No path found between the channels.
BFFA2012	Beginning and end of adjacent legs are not the same.
BFFA2013	Path contains a leg with un-connectable channels.
BFFA2014	A leg in the path is already connected.
BFFA2015	A channel cannot be connected to itself.

IVI Engine Status Codes:

Status	Description
ERRORS:	
BFFA0001	Instrument error. Call dl50Sw_error_query.
BFFA0002	Cannot open file.
BFFA0003	Error reading from file.

BFFA0004	Error writing to file.
BFFA0005	Driver module file not found.
BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.
BFFA000C	Invalid attribute.
BFFA000D	IVI attribute is not writable.
BFFA000E	IVI attribute is not readable.
BFFA000F	Invalid parameter.
BFFA0010	Invalid value.
BFFA0011	Function not supported.
BFFA0012	Attribute not supported.
BFFA0013	Value not supported.
BFFA0014	Invalid type.
BFFA0015	Types do not match.
BFFA0016	Attribute already has a value waiting to be updated.
BFFA0017	Specified item already exists.
BFFA0018	Not a valid configuration.
BFFA0019	Requested item does not exist or value not available.
BFFA001A	Requested attribute value not known.
BFFA001B	No range table.
BFFA001C	Range table is invalid.
BFFA001D	Object or item is not initialized.
BFFA001E	Non-interchangeable behavior.
BFFA001F	No channel table has been built for the session.
BFFA0020	Channel name specified is not valid.
BFFA0021	Unable to allocate system resource.
BFFA0022	Permission to access file was denied.
BFFA0023	Too many files are already open.
BFFA0024	Unable to create temporary file in target directory.
BFFA0025	All temporary filenames already used.
BFFA0026	Disk is full.
BFFA0027	Cannot find configuration file on disk.
BFFA0028	Cannot open configuration file.
BFFA0029	Error reading configuration file.
BFFA002A	Invalid ViInt32 value in configuration file.
BFFA002B	Invalid ViReal64 value in configuration file.
BFFA002C	Invalid ViBoolean value in configuration file.
BFFA002D	Entry missing from configuration file.
BFFA002E	Initialization failed in driver DLL.
BFFA002F	Driver module has unresolved external reference.
BFFA0030	Cannot find CVI Run-Time Engine.
BFFA0031	Cannot open CVI Run-Time Engine.
BFFA0032	CVI Run-Time Engine has invalid format.

BFFA0033	CVI Run-Time Engine is missing required function(s).
BFFA0034	CVI Run-Time Engine initialization failed.
BFFA0035	CVI Run-Time Engine has unresolved external reference.
BFFA0036	Failure loading CVI Run-Time Engine.
BFFA0037	Cannot open DLL for read exports.
BFFA0038	DLL file is corrupt.
BFFA0039	No DLL export table in DLL.
BFFA003A	Unknown attribute name in default configuration file.
BFFA003B	Unknown attribute value in default configuration file.
BFFA003C	Memory pointer specified is not known.
BFFA003D	Unable to find any channel strings.
BFFA003E	Duplicate channel string.
BFFA003F	Duplicate virtual channel name.
BFFA0040	Missing virtual channel name.
BFFA0041	Bad virtual channel name.
BFFA0042	Unassigned virtual channel name.
BFFA0043	Bad virtual channel assignment.
BFFA0044	Channel name required.
BFFA0045	Channel name not allowed.
BFFA0046	Attribute not valid for channel.
BFFA0047	Attribute must be channel based.
BFFA0048	Channel already excluded.
BFFA0049	Missing option name (nothing before the '=').
BFFA004A	Missing option value (nothing after the '=').
BFFA004B	Bad option name.
BFFA004C	Bad option value.
BFFA004D	Operation only valid on a class driver session.
BFFA004E	"ivi.ini" filename is reserved.
BFFA004F	Duplicate run-time configuration entry.
BFFA0050	Index parameter is one-based.
BFFA0051	Index parameter is too high.
BFFA0052	Attribute is not cacheable.
BFFA0053	You cannot export a ViAddr attribute to the end-user.
BFFA0054	Bad channel string in channel string list.
BFFA0055	Bad prefix name in default configuration file.

VISA Status Codes:

Status	**Description**
WARNINGS:	
3FFF0002	Event enabled for one or more specified mechanisms.
3FFF0003	Event disabled for one or more specified mechanisms.
3FFF0004	Successful, but queue already empty.
3FFF0005	Specified termination character was read.
3FFF0006	Number of bytes transferred equals input count.
3FFF0077	Configuration non-existent or could not be loaded.
3FFF007D	Open successful, but the device not responding.
3FFF0080	Wait successful, but more event objects available.

3FFF0082	Specified object reference is uninitialized.
3FFF0084	Attribute value not supported.
3FFF0085	Status code could not be interpreted.
3FFF0088	Specified I/O buffer type not supported.
3FFF0098	Successful, but invoke no handlers for this event.
3FFF0099	Successful but session has nested shared locks.
3FFF009A	Successful but session has nested exclusive locks.
3FFF009B	Successful but operation not asynchronous.

ERRORS:

BFFF0000	Unknown system error (miscellaneous error).
BFFF000E	Session or object reference is invalid.
BFFF000F	Resource is locked.
BFFF0010	Invalid expression specified for search.
BFFF0011	Resource is not present in the system.
BFFF0012	Invalid resource reference specified. Parsing error.
BFFF0013	Invalid access mode.
BFFF0015	Timeout expired before operation completed.
BFFF0016	Unable to deallocate session data structures.
BFFF001B	Specified degree is invalid.
BFFF001C	Specified job identifier is invalid.
BFFF001D	Attribute is not supported by the referenced object.
BFFF001E	Attribute state not supported by the referenced object.
BFFF001F	Specified attribute is read-only.
BFFF0020	Lock type lock not supported by this resource.
BFFF0021	Invalid access key.
BFFF0026	Specified event type not supported by the resource.
BFFF0027	Invalid mechanism specified.
BFFF0028	A handler was not installed.
BFFF0029	Handler reference either invalid or was not installed.
BFFF002A	Specified event context invalid.
BFFF002D	Event queue for specified type has overflowed.
BFFF002F	Event type must be enabled in order to receive.
BFFF0030	User abort during transfer.
BFFF0034	Violation of raw write protocol during transfer.
BFFF0035	Violation of raw read protocol during transfer.
BFFF0036	Device reported output protocol error during transfer.
BFFF0037	Device reported input protocol error during transfer.
BFFF0038	Bus error during transfer.
BFFF0039	Unable to queue asynchronous operation.
BFFF003A	Unable to start operation because setup is invalid.
BFFF003B	Unable to queue the asynchronous operation.
BFFF003C	Insufficient resources to perform memory allocation.
BFFF003D	Invalid buffer mask specified.
BFFF003E	I/O error.
BFFF003F	Format specifier invalid.
BFFF0041	Format specifier not supported.

BFFF0042	Trigger line is currently in use.
BFFF004A	Service request not received for the session.
BFFF004E	Invalid address space specified.
BFFF0051	Invalid offset specified.
BFFF0052	Invalid access width specified.
BFFF0054	Offset not accessible from this hardware.
BFFF0055	Source and destination widths are different.
BFFF0057	Session not currently mapped.
BFFF0059	Previous response still pending.
BFFF005F	No listeners condition detected.
BFFF0060	Interface not currently the controller in charge.
BFFF0061	Interface not the system controller.
BFFF0067	Session does not support this operation.
BFFF006A	A parity error occurred during transfer.
BFFF006B	A framing error occurred during transfer.
BFFF006C	An overrun error occurred during transfer.
BFFF0070	Offset not properly aligned for operation access width.
BFFF0071	Specified user buffer not valid.
BFFF0072	Resource valid, but VISA cannot access it.
BFFF0076	Width not supported by this hardware.
BFFF0078	Invalid parameter value, parameter unknown.
BFFF0079	Invalid protocol.
BFFF007B	Invalid window size.
BFFF0080	Session currently contains a mapped window.
BFFF0081	Operation not implemented.
BFFF0083	Invalid length.
BFFF0091	Invalid mode.
BFFF009C	Session did not have a lock on the resource.
BFFF009D	The device does not export any memory.
BFFF009E	VISA-required code library not located or not loaded.

VXIPnP Driver Status Codes:

Status	**Description**
WARNINGS:	
3FFC0101	Instrument does not have ID Query capability.
3FFC0102	Instrument does not have Reset capability.
3FFC0103	Instrument does not have Self-Test capability.
3FFC0104	Instrument does not have Error Query capability.
3FFC0105	Instrument does not have Revision Query capability.
ERRORS:	
BFFC0001	Parameter 1 out of range, or error trying to set it.
BFFC0002	Parameter 2 out of range, or error trying to set it.
BFFC0003	Parameter 3 out of range, or error trying to set it.
BFFC0004	Parameter 4 out of range, or error trying to set it.
BFFC0005	Parameter 5 out of range, or error trying to set it.
BFFC0006	Parameter 6 out of range, or error trying to set it.

BFFC0007	Parameter 7 out of range, or error trying to set it.
BFFC0008	Parameter 8 out of range, or error trying to set it.
BFFC0011	Instrument failed the ID Query.
BFFC0012	Invalid response from instrument.

errorMessage Returns the user-readable message string that corresponds to the status code you specify. You must pass a ViChar array with at least 256 bytes.

Return Value Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ ClearError function**.**

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_error_query

This function reads an error code and a message from the instrument's error queue.

PROTOTYPE: ViStatus dl50Sw_error_query (ViSession instrumentHandle, ViPInt32 errorCode, ViChar _VI_FAR errorMessage[]);

<int> = dl50Sw_error_query(ViSession instrumentHandle, errorcode, errorMessage[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

errorCode

Returns the error code read from the instrument's error queue.

errorMessage

Returns the error message string read from the instrument's error message queue. You must pass a ViChar array with at least 256 bytes.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetAttributeViBoolean

This function queries the value of a ViBoolean attribute. You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid.

PROTOTYPE: ViStatus dl50Sw_GetAttributeViBoolean (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViPBoolean attributeValue);

<int> = dl50Sw_GetAttributeViBoolean(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle
The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Read-only attributes appear dim in the list box. If you select a read-

only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViBoolean type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViBoolean are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetAttributeViInt32

This function queries the value of a ViInt32 attribute. You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid.

PROTOTYPE: ViStatus dl50Sw_GetAttributeViInt32 (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViPInt32 attributeValue);

<int> = dl50Sw_GetAttributeViInt32(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle
The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViInt32 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue

Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetAttributeViReal64

This function queries the value of a ViReal64 attribute. You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid.

PROTOTYPE: ViStatus dl50Sw_GetAttributeViReal64 (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViPReal64 attributeValue);

<int> = dl50Sw_GetAttributeViReal64(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName

If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViReal64 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViReal64 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue

Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetAttributeViSession

This function queries the value of a ViSession attribute. You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid.

PROTOTYPE: ViStatus dl50Sw_GetAttributeViSession (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViPSession attributeValue);

<int> dl50Sw_GetAttributeViSession(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName

If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViSession type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViSession are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue

Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetAttributeViString

This function queries the value of a ViString attribute. You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid.

You must provide a ViChar array to serve as a buffer for the value. You pass the number of bytes in the buffer as the Buffer Size parameter. If the current value of the attribute, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you want to call this function just to get the required buffer size, you can pass 0 for the Buffer Size and VI_NULL for the Attribute Value buffer.

If you want the function to fill in the buffer regardless of the number of bytes in the value, pass a negative number for the Buffer Size parameter.

PROTOTYPE: ViStatus dl50Sw_GetAttributeViString (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViInt32 bufferSize, ViChar _VI_FAR attributeValue[]);

<init> = dl50Sw_GetAttributeString(ViSession instrumentHandle, channelName[], attributeID, bufferSize, attributeValue[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName

If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViString type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViString are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

bufferSize

Pass the number of bytes in the ViChar array you specify for the Attribute Value parameter.

If the current value of the attribute, including the terminating NUL byte, contains more bytes that you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is

"123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI_NULL for the Attribute Value buffer parameter.

Default Value: 512

attributeValue

The buffer in which the function returns the current value of the attribute. The buffer must be of type ViChar and have at least as many bytes as indicated in the Buffer Size parameter.

If the current value of the attribute, including the terminating NUL byte, contains more bytes that you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you specify 0 for the Buffer Size parameter, you can pass VI_NULL for this parameter.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking on it or by selecting it and then pressing **<ENTER>**.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire

value. For example, if the value is “123456” and the Buffer Size is 4, the function places “123” into the buffer and returns 7.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetChannelName

This function returns the highest-level channel name that corresponds to the specific driver channel string that is in the channel table at an index you specify. By passing 0 for the buffer size, the caller can ascertain the buffer size required to get the entire channel name string and then call the function again with a sufficiently large buffer.

PROTOTYPE: ViStatus dl50Sw_GetChannelName (ViSession instrumentHandle, ViInt32 index, ViInt32 bufferSize, ViChar _VI_FAR channelName[]);

<int> = dl50Sw_GetChannelName(ViSession instrumentHandle, index, bufferSize, channelName[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

index

A 1-based index into the channel table.

bufferSize

Pass the number of bytes in the ViChar array you specify for the Channel Name parameter.

If the channel name, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies BufferSize - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is “123456” and the Buffer Size is 4, the function places “123” into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

f you pass 0, you can pass VI_NULL for the Channel Name buffer parameter.

channelName

Returns the highest-level channel name that corresponds to the specific driver channel string that is in the channel table at an index you specify..

The buffer must contain at least as many elements as the value you specify with the Buffer Size parameter. If the channel name description, including the terminating NUL byte, contains more bytes than you indicate with the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass 0 for the Buffer Size, you can pass VI_NULL for this parameter.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, or if the status code is not listed below, call the dl50Sw_error_message function. To obtain additional information about the error condition, use the dl50Sw_GetError and dl50Sw_ClearError functions.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetError

This function retrieves and then clears the IVI error information for the session or the current execution thread. One exception exists: If the BufferSize information. By passing 0 for the buffer size, the caller can ascertain the buffer size required to get the entire error description string and then call the function again with a sufficiently large buffer.

If the user specifies a valid IVI session for the InstrumentHandle parameter, Get Error retrieves and then clears the error information for the session. If the user passes VI_NULL for the InstrumentHandle parameter, this function retrieves and then clears the error information for the current execution thread. If the InstrumentHandle parameter is an invalid session, the function does nothing and returns an error. Normally, the error information describes the first error that occurred since the user last called dl50Sw_GetError or dl50Sw_ClearError.

PROTOTYPE: ViStatus dl50Sw_GetError (ViSession instrumentHandle, ViPStatus code, ViInt32 bufferSize, ViChar _VI_FAR description[]);

<int> = dl50Sw_GetError(ViSession instrumentHandle, code, buffersize, description[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

code

Returns the error code for the session or execution thread. If you pass 0 for the Buffer Size, you can pass VI_NULL for this parameter.

bufferSize

If the error description, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies BufferSize - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is “123456” and the Buffer Size is 4, the function places “123” into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI_NULL for the Description buffer parameter.

description

Returns the error description for the IVI session or execution thread. If there is no description, the function returns an empty string.

The buffer must contain at least as many elements as the value you specify with the Buffer Size parameter. If the error description, including the terminating NUL byte, contains more bytes than you indicate with the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass 0 for the Buffer Size, you can pass VI_NULL for this parameter.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetNextCoercionRecord

This function returns the coercion information associated with the IVI session. This function retrieves and clears the oldest instance in which the instrument driver coerced a value you specified to another value.

If you set the DL50SW_ATTR_RECORD_COERCIONS attribute to VI_TRUE, the instrument driver keeps a list of all coercions it makes on ViInt32 or ViReal64 values you pass to instrument driver functions. You use this function to retrieve information from that list.

If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is “123456” and the Buffer Size is 4, the function places “123” into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI_NULL for the Coercion Record buffer parameter.

The function returns an empty string in the Coercion Record parameter if no coercion records remain for the session.

PROTOTYPE: ViStatus dl50Sw_GetNextCoercionRecord (ViSession instrumentHandle, ViInt32 bufferSize, ViChar _VI_FAR coercionRecord[]);

<int> = dl50Sw_GetNextCoercionRecord(ViSession instrumentHandle, bufferSize, coercionRecord[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init function. The handle identifies a particular instrument session.

bufferSize

Pass the number of bytes in the ViChar array you specify for the Coercion Record parameter. If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI_NULL for the Coercion Record buffer parameter.

coercionRecord

Returns the next coercion record for the IVI session. If there are no coercion records, the function returns an empty string.

The buffer must contain at least as many elements as the value you specify with the Buffer Size parameter. If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate with the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

This parameter returns an empty string if no coercion records remain for the session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetNextInterchangeWarning

This function returns the interchangeability warnings associated with the IVI session. It retrieves and clears the oldest instance in which the class driver recorded an interchangeability warning. Interchangeability warnings indicate that using your application with a different instrument might cause different behavior. You use this function to retrieve interchangeability warnings.

The driver performs interchangeability checking when the DL50SW_ATTR_ INTERCHANGE_CHECK attribute is set to VI_TRUE.

The function returns an empty string in the Interchange Warning parameter if no interchangeability warnings remain for the session.

In general, the instrument driver generates interchangeability warnings when an attribute that affects the behavior of the instrument is in a state that you did not specify.

PROTOTYPE: ViStatus dl50Sw_GetNextInterchangeWarning (ViSession instrumentHandle, ViInt32 bufferSize, ViChar _VI_FAR interchangeWarning[]);

<int> = dl50Sw_GetNextInterchangeWarning(ViSession instrumentHandle, bufferSize, interchangeWarning[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

bufferSize

Pass the number of bytes in the ViChar array you specify for the Interchange Warning parameter.

If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is “123456” and the Buffer Size is 4, the function places “123” into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI_NULL for the Interchange Warning buffer parameter.

interchangeWarning

Returns the next interchange warning for the IVI session. If there are no interchange warnings, the function returns an empty string.

The buffer must contain at least as many elements as the value you specify with the Buffer Size parameter. If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate with the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

This parameter returns an empty string if no interchangeability warnings remain for the session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, or if the status code is not listed below, call the dl50Sw_error_message function. To obtain additional information about the error condition, use the dl50Sw_GetError and dl50Sw_ClearError functions.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_GetPath

In some cases there is more than one possible path between two channels. The driver or the instrument selects the path when you connect two channels with the dl50Sw_Connect function. Thus, you cannot guarantee that every call to the dl50Sw_Connect function establishes exactly the same path when you pass the same channels. This function returns a string that uniquely identifies the path you create with the dl50Sw_Connect function. You can pass this string to the dl50Sw_SetPath function to establish the exact same path in the future.

Note: This function returns only those paths that you explicitly create by calling dl50Sw_Connect and dl50Sw_SetPath functions. For example, if you connect channels CH1 and CH3, and then channels CH2 and CH3, the explicit path between channels CH1 and Ch2 does not exist and this function returns an error.

PROTOTYPE: ViStatus dl50Sw_GetPath (ViSession instrumentHandle, ViChar _VI_FAR channel1[], ViChar _VI_FAR channel2[], ViInt32 bufferSize, ViChar _VI_FAR path[]);

<int> = dl50Sw_GetPath(ViSession instrumentHandle, channel1[], channel2[], bufferSize, path[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channel1

You identify a path with two channels. Pass one of the channel names for which you want to obtain a path. Pass the other channel name as the Channel 2 parameter.

channel2

You identify a path with two channels. Pass one of the channel names for which you want to obtain a path. Pass the other channel name as the Channel 1 parameter.

bufferSize

Pass the number of bytes in the ViChar array you specify for the Path List parameter.

If the current value of the attribute, including the terminating NUL byte, contains more bytes that you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "R1->C1" and the Buffer Size is 4, the function places "R1-" into the buffer and returns 7.

If you pass 0, you can pass VI_NULL for the Path parameter. Thisenables you to find out the path size and to allocate the buffer of the appropriate size before calling this function again.

path

The comma-separated path between channels you specify in the Channel 1 and Channel 2 parameters.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_init

This function performs the following initialization actions:

- Creates a new IVI instrument driver session.

- Opens a session to the specified device using the interface and address you specify for the Resource Name parameter.

- If the ID Query parameter is set to VI_TRUE, this function queries the instrument ID and checks that it is valid for this instrument driver.

- If the Reset parameter is set to VI_TRUE, this function resets the instrument to a known state.

- Sends initialization commands to set the instrument to the state necessary for the operation of the instrument driver.

- Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Note: This function creates a new session each time you invoke it. Although you can open more than one IVI session for the same resource, it is best not to do so. You can use the same session in multiple program threads. You can use functions dl50Sw_LockSession and dl50Sw_UnlockSession to protect sections of code that require exclusive access to the resource.

PROTOTYPE: ViStatus dl50Sw_init (ViRsrc resourceName, ViBoolean IDQuery, ViBoolean resetDevice, ViPSession instrumentHandle);

<int> = dl50Sw_init(resourceName, IDQuery, resetDevice,

resourceName

Pass the resource name of the device to initialize. You can also pass the name of a virtual instrument or logical name that you configure with the IVI Configuration utility. The virtual instrument identifies a specific device and specifies the initial settings for the session. A logical Name identifies a particular virtual instrument.

Note: Refer to the following table below for the exact grammar to use for this parameter. Optional fields are shown in square brackets ([]).

Syntax
[search path]2050::<slot number>::INSTR
<LogicalName>
<DriverSession>

If you do not specify a value for an optional field, the following values are used:

Optional Field - Value
search path - none

The following table contains example valid values for this parameter.

"Valid Value" - Description
"2050::9::INSTR"
- 2050 platform
- Slot 9 of the Resource Interface Chassis

"visa://DigalogPC_2050/2050::9::INSTR"
- 2050 platform connected to the network
- 2050 PC's name: DigalogPC_2050
- Slot 9 of the Resource Interface Chassis

"AuxCard1" - Logical name "AuxCard1"

Default Value: "2050::10::INSTR"

IDQuery

Specify whether you want the instrument driver to perform an ID Query.

Valid Range:
VI_TRUE (1) - Perform ID Query (Default Value)
VI_FALSE (0) - Skip ID Query

Note: When you set this parameter to VI_TRUE, the driver verifies that the instrument you initialize is a type that this driver supports.

Circumstances can arise where it is undesirable to send an ID Query command string to the instrument. When you set this parameter to VI_FALSE, the function initializes the instrument without performing an ID Query.

resetDevice

Specify whether you want the to reset the instrument during the initialization procedure.

Valid Range:
VI_TRUE (1) - Reset Device (Default Value)
VI_FALSE (0) - Don't Reset

instrumentHandle

Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Notes:

(1) This function creates a new session each time you invoke it. This is useful if you have multiple physical instances of the same type of instrument.

(2) Avoid creating multiple concurrent sessions to the same physical instrument. Although you can create more than one IVI session for the same resource, it is best not to do so. A better approach is to use the same IVI session in multiple execution threads. You can use functions dl50Sw_LockSession and dl50Sw_UnlockSession to protect sections of code that require exclusive access to the resource.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_InitWithOptions

This function performs the following initialization actions:

- Creates a new IVI instrument driver and optionally sets the initial state of the following session attributes:

DL50SW_ATTR_RANGE_CHECK
DL50SW_ATTR_QUERY_INSTRUMENT_STATUS
DL50SW_ATTR_CACHE
DL50SW_ATTR_SIMULATE
DL50SW_ATTR_RECORD_COERCIONS

- Opens a session to the specified device using the interface and address you specify for the Resource Name parameter.

- If the ID Query parameter is set to VI_TRUE, this function queries the instrument ID and checks that it is valid for this instrument driver.

- If the Reset parameter is set to VI_TRUE, this function resets the instrument to a known state.

- Sends initialization commands to set the instrument to the state necessary for the operation of the instrument driver.

- Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Note: This function creates a new session each time you invoke it. Although you can open more than one IVI session for the same resource, it is best not to do so. You can use the same session in multiple program threads. You can use the dl50Sw_LockSession and dl50Sw_UnlockSession functions to protect sections of code that require exclusive access to the resource.

PROTOTYPE: ViStatus dl50Sw_InitWithOptions (ViRsrc resourceName, ViBoolean IDQuery, ViBoolean resetDevice, ViChar _VI_FAR optionString[], ViPSession instrumentHandle);

<int> = dl50Sw_initWithOptions(resourceName, IDQuery, ViBoolean, optionString[], instrumentHandle);

resourceName

Pass the resource name of the device to initialize. You can also pass the name of a virtual instrument or logical name that you configure with the IVI Configuration utility. The virtual instrument identifies a specific device and specifies the initial settings for the session. A logical Name identifies a particular virtual instrument.

Note: Refer to the following table below for the exact grammar to use for this parameter. Optional fields are shown in square brackets ([]).

Syntax
[search path]2050::<slot number>::INSTR
<LogicalName>
<DriverSession>

If you do not specify a value for an optional field, the following values are used:

Optional Field - Value
search path - none

The following table contains example valid values for this parameter.

"Valid Value" - Description
"2050::9::INSTR"
- 2050 platform
- Slot 9 of the Resource Interface Chassis

"visa://DigalogPC_2050/2050::9::INSTR"
- 2050 platform connected to the network
- 2050 PC's name: DigalogPC_2050
- Slot 9 of the Resource Interface Chassis

"AuxCard1" - Logical name "AuxCard1"

Default Value: "2050::10::INSTR"

IDQuery

Specify whether you want the instrument driver to perform an ID Query.

Valid Range:
VI_TRUE (1) - Perform ID Query (Default Value)
VI_FALSE (0) - Skip ID Query

Note: When you set this parameter to VI_TRUE, the driver verifies that the instrument you initialize is a type that this driver supports.

Circumstances can arise where it is undesirable to send an ID Query command string to the instrument. When you set this parameter to VI_FALSE, the function initializes the instrument without performing an ID Query.

resetDevice

Specify whether you want the to reset the instrument during the initialization procedure.

Valid Range:
VI_TRUE (1) - Reset Device (Default Value)
VI_FALSE (0) - Don't Reset

optionString

You can use this control to set the initial value of certain attributes for the session. The following table lists the attributes and the name you use in this parameter to identify the attribute.

Name	Attribute Defined Constant
RangeCheck	DL50SW_ATTR_RANGE_CHECK
QueryInstrStatus	DL50SW_ATTR_QUERY_INSTRUMENT_STATUS
Cache	DL50SW_ATTR_CACHE
Simulate	DL50SW_ATTR_SIMULATE
RecordCoercions	DL50SW_ATTR_RECORD_COERCIONS
DriverSetup	IVI_ATTR_DRIVER_SETUP

Note: The format of this string is, "AttributeName=Value" where AttributeName is the name of the attribute and Value is the value to which separate their assignments with a comma.

If you pass NULL or an empty string for this parameter and a VISA resource descriptor for the Resource Name parameter, the session uses the default values for the attributes. The default values for the attributes are shown on the next page:

Attribute Name	Default Value
RangeCheck	VI_TRUE
QueryInstrStatus	VI_FALSE
Cache	VI_TRUE
Simulate	VI_FALSE
RecordCoercions	VI_FALSE
DriverSetup	[see below]

If you pass NULL or an empty string for this parameter and a virtual instrument or logical name for the Resource Name parameter, the session uses the values that you configure for virtual instrument or logical name with the IVI Configuration utility.

You can override the values of the attributes by assigning a value explicitly in a string you pass for this parameter. You do not have to specify all of the attributes and may leave any of them out. If you do not specify one of the attributes, its default value or the value that you configure with the IVI Configuration utility will be used.

The following are the valid values for ViBoolean attributes:

True: 1, TRUE, or VI_TRUE
False: 0, False, or VI_FALSE

DriverSetup

The only tag recognized in the DriverSetup attribute is the "Model" tag. The value of this tag is used to define which model card the driver should simulate. If Simulate=0 this tag's value is ignored.

The following are the valid Model tag values for the DriverSetup attribute:

Model:1510

Example: DriverSetup=Model:1510

Default Value:
"Simulate=0,RangeCheck=1,QueryInstrStatus=0,Cache=1,
DriverSetup=Model:1510"

instrumentHandle

Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Notes:

(1) This function creates a new session each time you invoke it. This is useful if you have multiple physical instances of the same type of instrument.

(2) Avoid creating multiple concurrent sessions to the same physical instrument. Although you can create more than one IVI session for the same resource, it is best not to do so. A better approach is to use the same IVI session in multiple execution threads. You can use functions dl50Sw_LockSession and dl50Sw_UnlockSession to protect sections of code that require exclusive access to the resource.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_InvalidateAllAttributes

This function invalidates the cached values of all attributes for the session.

PROTOTYPE: ViStatus dl50Sw_InvalidateAllAttributes (ViSession instrumentHandle);

<int> = dl50Sw_InvalidateAllAttributes(ViSession instrumentHandle);

instrumentHandle

Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Notes:

(1) This function creates a new session each time you invoke it. This is useful if you have multiple physical instances of the same type of instrument.

(2) Avoid creating multiple concurrent sessions to the same physical instrument. Although you can create more than one IVI session for the same resource, it is best not to do so. A better approach is to use the same IVI session in multiple execution threads. You can use functions dl50Sw_LockSession and dl50Sw_UnlockSession to protect sections of code that require exclusive access to the resource.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_IsDebounced

This function returns the state of the switch module. It indicates if all the paths that you created have settled.

PROTOTYPE: ViStatus dl50Sw_IsDebounced (ViSession instrumentHandle, ViPBoolean isDebounced);

<int> = dl50Sw_IsDebounced(ViSession instrumentHandle, isDebounced);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

isDebounced

Indicates the state of the switch module. The driver returns the value of DL50SW_ATTR_IS_DEBOUNCED attribute.

The value VI_TRUE indicates that all the paths that you created have settled.

The value VI_FALSE indicates that all the paths that you created have not settled.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_LockSession

This function obtains a multithread lock on the instrument session. Before it does so, it waits until all other execution threads have released their locks on the instrument session.

Other threads might have obtained a lock on this session in the following ways:

- The user's application called dl50Sw_LockSession.

- A call to the instrument driver locked the session.

- A call to the IVI engine locked the session.

After your call to dl50Sw_LockSession returns successfully, no other threads can access the instrument session until you call dl50Sw_UnlockSession.

Use dl50Sw_LockSession and dl50Sw_UnlockSession around a sequence of calls to instrument driver functions if you require that the instrument retain its settings through the end of the sequence.

You can safely make nested calls to dl50Sw_LockSession within the same thread. To completely unlock the session, you must balance each call to use the Caller Has Lock parameter in all calls to dl50Sw_LockSession and only once within the function regardless of the number of calls you make to dl50Sw_LockSession. This allows you to call dl50Sw_UnlockSession just once at the end of the function.

PROTOTYPE: ViStatus dl50Sw_LockSession (ViSession instrumentHandle, ViPBoolean callerHasLock);

<int> = dl50Sw_LockSession(ViSession instrumentHandle, callerHasLock);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

callerHasLock

This parameter serves as a convenience. If you do not want to use this parameter, pass VI_NULL.

Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to VI_FALSE. Pass the address of the same local variable to any other calls you make to dl50Sw_LockSession or dl50Sw_UnlockSession in the same function.

The parameter is an input/output parameter. dl50Sw_LockSession and dl50Sw_UnlockSession each inspect the current value and take the following actions:

- If the value is VI_TRUE, dl50Sw_LockSession does not lock the session again. If the value is VI_FALSE, dl50Sw_LockSession obtains the lock and sets the value of the parameter to VI_TRUE.

- If the value is VI_FALSE, dl50Sw_UnlockSession does not attempt to unlock the session. If the value is VI_TRUE, dl50Sw_UnlockSession releases the lock and sets the value of the parameter to VI_FALSE.

Thus, you can, call dl50Sw_UnlockSession at the end of your function without worrying about whether you actually have the lock.

Example:

```
ViStatus TestFunc (ViSession vi, ViInt32 flags)
{
   ViStatus error = VI_SUCCESS;
   ViBoolean haveLock = VI_FALSE;

   if (flags & BIT_1)
      {
      viCheckErr( dl50Sw_LockSession(vi, &haveLock));
      viCheckErr( TakeAction1(vi));
      if (flags & BIT_2)
         {
         viCheckErr( dl50Sw_UnlockSession(vi, &haveLock));
         viCheckErr( TakeAction2(vi));
```

```
            viCheckErr( dl50Sw_LockSession(vi, &haveLock);
            }
        if (flags & BIT_3)
            viCheckErr( TakeAction3(vi));
        }

    Error:
        /*
          At this point, you cannot really be sure that
          you have the lock.  Fortunately, the haveLock
          variable takes care of that for you.
        */
        dl50Sw_UnlockSession(vi, &haveLock);
        return error;
    }
```

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_reset

This function resets the instrument to a known state and sends initialization commands to the instrument. The initialization commands set instrument settings such as Headers Off, Short Command form, and Data Transfer Binary to the state necessary for the operation of the instrument driver.

PROTOTYPE: ViStatus dl50Sw_reset (ViSession instrumentHandle);

<int> = dl50Sw_reset(ViSession instrumentHandle);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_ResetInterchangeCheck

When developing a complex test system that consists of multiple test modules, it is generally a good idea to design the test modules so that they can run in any order. To do so requires ensuring that each test module completely configures the state of each instrument it uses. If a particular test module does not completely configure the state of an instrument, the state of the instrument depends on the configuration from a previously executed test module. If you execute the test modules in a different order, the behavior of the instrument and therefore the entire test module is likely to change. This change in behavior is generally instrument specific and represents an interchangeability problem.

You can use this function to test for such cases. After you call this function, the interchangeability checking algorithms in the specific driver ignore all previous configuration operations. By calling this function at the beginning of a test module, you can determine whether the test module has dependencies on the operation of previously executed test modules.

This function does not clear the interchangeability warnings from the list of previously recorded interchangeability warnings. If you want to guarantee that the dl50Sw_GetNextInterchangeWarning function only returns those interchangeability warnings that are generated after calling this function, you must clear the list of interchangeability warnings. You can clear the interchangeability warnings list by repeatedly calling the dl50Sw_GetNextInterchangeWarning function until no more interchangeability warnings are returned. If you are not interested in the content of those warnings, you can call the dl50Sw_ClearInterchangeWarnings function.

PROTOTYPE: ViStatus dl50Sw_ResetInterchangeCheck (ViSession instrumentHandle);

<int> = dl50Sw_ResetInterchangeCheck(ViSession instrumentHandle);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_ResetWithDefaults

This function resets the instrument and applies initial user-specified settings from the Logical Name which was used to initialize the session. If the session was created without a Logical Name, this function is equivalent to the dl50Sw_reset function.

PROTOTYPE: ViStatus dl50Sw_ResetWithDefaults (ViSession instrumentHandle);

<int> = dl50Sw_ResetWithDefaults(ViSession instrumentHandle);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_revision_query

This function returns the revision numbers of the instrument driver and instrument firmware.

PROTOTYPE: ViStatus dl50Sw_revision_query (ViSession instrumentHandle, ViChar _VI_FAR instrumentDriverRevision[], ViChar _VI_FAR firmwareRevision[]);

<int> = dl50Sw_revision_query(ViSession instrumentHandle, instrumentDriverRevision[], firmwareRevision[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

instrumentDriverRevision

Returns the instrument driver software revision numbers in the form of a string. You must pass a ViChar array with at least 256 bytes.

firmwareRevision

The firmware revision is not available at this time. A blank string will be returned for the firmware revision. You must pass a ViChar array with at least 256 bytes.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_self_test

This function runs the instrument's self test routine and returns the test result(s).

PROTOTYPE: ViStatus dl50Sw_self_test (ViSession instrumentHandle, ViPInt16 selfTestResult, ViChar _VI_FAR selfTestMessage[]);

<int> = dl50Sw_ViSession instrumentHandlee, selfTestResult, selfTestMessage[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

selfTestResult

This control contains the value returned from the instrument self test. Zero means success. For any other code, see the device's operator's manual.

Self-Test Code	Description
0	Passed self test
1	Self test failed

selfTestMessage

Returns the self-test response string from the instrument. See the device's operation manual for an explanation of the string's contents. You must pass a ViChar array with at least 256 bytes.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_SetAttributeViBoolean

This function sets the value of a ViBoolean attribute. This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

PROTOTYPE: ViStatus dl50Sw_SetAttributeViBoolean (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViBoolean attributeValue);

<int> = dl50Sw_SetAttributeViBoolean(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName

If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViBoolean type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViBoolean are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking

on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_SetAttributeViInt32

This function sets the value of a ViInt32 attribute. This is a low-level function that you can use to set the values of instrument-specific attributes and attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

PROTOTYPE: ViStatus dl50Sw_SetAttributeViInt32 (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViInt32 attributeValue);

<int> = dl50Sw_SetAttributeViInt32(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName

If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViInt32 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking

on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_SetAttributeViReal64

This function sets the value of a ViReal64 attribute. This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

PROTOTYPE: ViStatus dl50Sw_SetAttributeViReal64 (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViReal64 attributeValue);

<int> = dl50Sw_SetAttributeViReal64(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName

If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViReal64 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViReal64 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking

on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_SetAttributeViSession

This function sets the value of a ViSession attribute. This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

PROTOTYPE: ViStatus dl50Sw_SetAttributeViSession (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViSession attributeValue);

<int> = dl50Sw_SetAttributeViSession(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName

If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViSession type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViSession are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking

on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_SetAttributeViString

This function sets the value of a ViString attribute. This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.

- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

PROTOTYPE: ViStatus dl50Sw_SetAttributeViString (ViSession instrumentHandle, ViChar _VI_FAR channelName[], ViAttr attributeID, ViChar _VI_FAR attributeValue[]);

<int> = dl50Sw_SetAttributeViString(ViSession instrumentHandle, channelName[], attributeID, attributeValue);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

channelName

If the attribute is channel-based, this parameter specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string.

attributeID

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

Click on the control or press **<ENTER>**, **<spacebar>**, or **<ctrl-down arrow>**, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing **<ENTER>**.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViStringn type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViString are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

If you want to enter a variable name, press **<CTRL-T>** to change this ring control to a manual input box.

If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing **<ENTER>**.

attributeValue

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing **<ENTER>** on this control. Select a value by double-clicking

on it or by selecting it and then pressing **<ENTER>**.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_SetPath

This function connects two channels by establishing the exact path you specify with the pathList parameter.

PROTOTYPE: ViStatus dl50Sw_SetPath (ViSession instrumentHandle, ViChar _VI_FAR pathList[]);

<int> = dl50Sw_SetPath(ViSession instrumentHandle, pathList[]);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

pathList

Pass the path list for the path you previously created that you want the switch module to establish. You obtain the path list for a path you previously created with the dl50Sw_GetPath function.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_UnlockSession

This function releases a lock that you acquired on an instrument session using dl50Sw_LockSession. Refer to dl50Sw_LockSession for additional information on session locks.

PROTOTYPE: ViStatus dl50Sw_UnlockSession (ViSession instrumentHandle, ViPBoolean callerHasLock);

<int> = dl50Sw_UnlockSession(ViSession instrumentHandle, callerHasLock);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

callerHasLock

This parameter serves as a convenience. If you do not want to use this parameter, pass VI_NULL.

Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to VI_FALSE. Pass the address of the same local variable to any other calls you make to dl50Sw_LockSession or dl50Sw_UnlockSession in the same function.

The parameter is an input/output parameter. dl50Sw_LockSession and dl50Sw_UnlockSession each inspect the current value and take the following actions:

- If the value is VI_TRUE, dl50Sw_LockSession does not lock the session again. If the value is VI_FALSE, dl50Sw_LockSession obtains the lock and sets the value of the parameter to VI_TRUE.

- If the value is VI_FALSE, dl50Sw_UnlockSession does not attempt to unlock the session. If the value is VI_TRUE, dl50Sw_UnlockSession releases the lock and sets the value of the parameter to VI_FALSE.

Thus, you can, call dl50Sw_UnlockSession at the end of your function without worrying about whether you actually have the lock.

Example:

```
ViStatus TestFunc (ViSession vi, ViInt32 flags)
{
   ViStatus error = VI_SUCCESS;
   ViBoolean haveLock = VI_FALSE;

   if (flags & BIT_1)
      {
      viCheckErr( dl50Sw_LockSession(vi, &haveLock));
      viCheckErr( TakeAction1(vi));
      if (flags & BIT_2)
         {
         viCheckErr( dl50Sw_UnlockSession(vi, &haveLock));
         viCheckErr( TakeAction2(vi));
         viCheckErr( dl50Sw_LockSession(vi, &haveLock);
         }
      if (flags & BIT_3)
         viCheckErr( TakeAction3(vi));
      }

Error:
   /*
     At this point, you cannot really be sure that
     you have the lock.  Fortunately, the haveLock
     variable takes care of that for you.
   */
   dl50Sw_UnlockSession(vi, &haveLock);
   return error;
}
```

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

dl50Sw_WaitForDebounce

This function returns after all the paths that you create have settled.

PROTOTYPE: ViStatus dl50Sw_WaitForDebounce (ViSession instrumentHandle, ViInt32 maximumTime_ms);

<int> = dl50Sw_WaitForDebounce(ViSession instrumentHandle, maximumTime_ms);

instrumentHandle

The ViSession handle that you obtain from the dl50Sw_init or dl50Sw_InitWithOptions function. The handle identifies a particular instrument session.

maximumTime_ms

Specifies the maximum length of time for this function to wait until all switches in the switch module debounce. If the time you specify elapses before all switches debounce, this function returns a timeout error. The units are milliseconds.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the dl50Sw_error_message function. To obtain additional information about the error condition, call the dl50Sw_GetError function. To clear the error information from the driver, call the dl50Sw_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This instrument driver returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include filenames that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviSwtch Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviSwtch Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

Attribute Information for the Following Functions:

dl50Sw_SetAttributeViInt32
dl50Sw_GetAttributeViInt32
dl50Sw_CheckAttributeViInt32
dl50Sw_SetAttributeViReal64
dl50Sw_GetAttributeViReal64
dl50Sw_CheckAttributeViReal64
dl50Sw_SetAttributeViSession
dl50Sw_GetAttributeViSession
dl50Sw_CheckAttributeViSession
dl50Sw_SetAttributeViBoolean
dl50Sw_GetAttributeViBoolean
dl50Sw_CheckAttributeViBoolean
dl50Sw_SetAttributeViString
dl50Sw_GetAttributeViString
dl50Sw_CheckAttributeViString

Inherent IVI Attributes

User Options

Range Check	DL50SW_ATTR_RANGE_CHECK
Query Instrument Status	DL50SW_ATTR_QUERY_INSTRUMENT_STATUS
Cache	DL50SW_ATTR_CACHE
Simulate	DL50SW_ATTR_SIMULATE
Record Value Coercions	DL50SW_ATTR_RECORD_COERCIONS
Interchange Check	DL50SW_ATTR_INTERCHANGE_CHECK

Driver Identification

Description
DL50SW_ATTR_SPECIFIC_DRIVER_DESCRIPTION

Driver Prefix	DL50SW_ATTR_SPECIFIC_DRIVER_PREFIX
Driver Vendor	DL50SW_ATTR_SPECIFIC_DRIVER_VENDOR
Revision	DL50SW_ATTR_SPECIFIC_DRIVER_REVISION

Class Specification Major Version
DL50SW_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION
Class Specification Minor Version
DL50SW_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION

Driver Capabilities

Supported Instrument Models
DL50SW_ATTR_SUPPORTED_INSTRUMENT_MODELS
Class Group Capabilities DL50SW_ATTR_GROUP_CAPABILITIES

Instrument Identification

Manufacturer
DL50SW_ATTR_INSTRUMENT_MANUFACTURER
Model
DL50SW_ATTR_INSTRUMENT_MODEL
Firmware Revision
DL50SW_ATTR_INSTRUMENT_FIRMWARE_REVISION

Advanced Session Information

Logical Name DL50SW_ATTR_LOGICAL_NAME
I/O Resource Descriptor
DL50SW_ATTR_IO_RESOURCE_DESCRIPTOR
Driver Setup DL50SW_ATTR_DRIVER_SETUP

Channel Configuration

Is Source Channel DL50SW_ATTR_IS_SOURCE_CHANNEL
Is Configuration Channel
DL50SW_ATTR_IS_CONFIGURATION_CHANNEL

Module Characteristics

Number of relays DL50SW_ATTR_RELAY_COUNT
Type of relay card DL50SW_ATTR_CARD_TYPE
Is Debounced DL50SW_ATTR_IS_DEBOUNCED
Settling Time DL50SW_ATTR_SETTLING_TIME
Bandwidth DL50SW_ATTR_BANDWIDTH
Maximum DC Voltage DL50SW_ATTR_MAX_DC_VOLTAGE
Maximum AC Voltage DL50SW_ATTR_MAX_AC_VOLTAGE
Maximum Switching DC Current
DL50SW_ATTR_MAX_SWITCHING_DC_CURRENT
Maximum Switching AC Current
DL50SW_ATTR_MAX_SWITCHING_AC_CURRENT
Maximum Carry DC Current
DL50SW_ATTR_MAX_CARRY_DC_CURRENT

Maximum Carry AC Current
DL50SW_ATTR_MAX_CARRY_AC_CURRENT
Maximum Switching DC Power
DL50SW_ATTR_MAX_SWITCHING_DC_POWER
Maximum Switching AC Power
DL50SW_ATTR_MAX_SWITCHING_AC_POWER
Maximum Carry DC Power
DL50SW_ATTR_MAX_CARRY_DC_POWER
Maximum Carry AC Power
DL50SW_ATTR_MAX_CARRY_AC_POWER
Characteristic Impedance
DL50SW_ATTR_CHARACTERISTIC_IMPEDANCE

Matrix Configuration

Number of Rows	DL50SW_ATTR_NUM_OF_ROWS
Number of Columns	DL50SW_ATTR_NUM_OF_COLUMNS
Number of Analog channels	DL50SW_ATTR_NUM_OF_ANALOG
Number of Digital channels	DL50SW_ATTR_NUM_OF_DIGITAL
Wire mode	DL50SW_ATTR_WIRE_MODE

Miscellaneous Attributes

Manufacturer ID	DL50SW_ATTR_MANUFACTURER_ID
Model Code	DL50SW_ATTR_MODEL_CODE

Hidden Attributes (not user-viewable)

Slot number of the RIC board	DL50SW_ATTR_SLOT_NUM

DL50SW_ATTR_BANDWIDTH

This channel-based attribute returns the bandwidth for the channel. The units are hertz.

DL50SW_ATTR_CACHE

This attribute specifies whether to cache the value of attributes. When caching is enabled, the instrument driver keeps track of the current instrument settings and avoids sending redundant commands to the instrument. Thus, you can significantly increase execution speed. The instrument driver can choose always to cache or never to cache particular attributes regardless of the setting of this attribute. The default value is VI_TRUE. Use the dl50Sw_ InitWithOptions function to override this value.

DL50SW_ATTR_CARD_TYPE
The relay card will either be a switching card or a matrix card. Values:

DL50SW_VAL_SWITCHING_TYPE	0
DL50SW_VAL_MATRIX_TYPE	1

DL50SW_ATTR_CHARACTERISTIC_IMPEDANCE
This channel-based attribute returns the characteristic impedance for the channel. The units are ohms.

DL50SW_ATTR_DRIVER_SETUP
Some cases exist where you must specify instrument driver options at initialization time. An example of this is specifying a particular instrument model from among a family of instruments that the driver supports. This is useful when using simulation. You can specify driver-specific options through the DriverSetup keyword in the optionsString parameter to the dl50Sw_InitWithOptions function. If you open an instrument using a logical name, you can also specify the options through the IVI Configuration Utility. The default value is an empty string.

DL50SW_ATTR_GROUP_CAPABILITIES
A string that contains a comma-separated list of class-extension groups that this driver implements.

DL50SW_ATTR_INSTRUMENT_FIRMWARE_REVISION
A string that contains the firmware revision information for the instrument you are currently using.

DL50SW_ATTR_INSTRUMENT_MANUFACTURER
A string that contains the name of the instrument manufacturer you are currently using.

DL50SW_ATTR_INSTRUMENT_MODEL
A string that contains the model number or name of the instrument that you are currently using.

DL50SW_ATTR_INTERCHANGE_CHECK
Specifies whether to perform interchangeability checking and retrieve interchangeability warnings. The default value is VI_FALSE.

Interchangeability warnings indicate that using your application with a different instrument might cause different behavior. You call dl50Sw_GetNextInterchangeWarning to extract interchange warnings. Call the dl50Sw_ClearInterchangeWarnings function to clear the list of interchangeability warnings without reading them. Interchangeability checking logs a warning for each attribute you have not set that affects the behavior of the instrument.

DL50SW_ATTR_IO_RESOURCE_DESCRIPTOR
Indicates the resource descriptor the driver uses to identify the physical device. If you initialize the driver with a logical name, this attribute contains the resource descriptor that corresponds to the entry in the IVI Configuration utility. If you initialize the instrument driver with the resource descriptor, this attribute contains that value.

DL50SW_ATTR_IS_CONFIGURATION_CHANNEL
This channel-based attribute specifies whether to reserve the channel for internal path creation. A channel that is available for internal path creation is called a configuration channel. The driver may use configuration channels to create paths between two channels you specify in the dl50Sw_Connect function. Configuration channels are not available for external connections. Set this attribute to VI_TRUE to mark the channel as a configuration channel. Set this attribute to VI_FALSE to mark the channel as available for external connections. After you identify a channel as a configuration channel, you cannot use that channel for external connections. The dl50Sw_Connect function returns the DL50SW_ERROR_IS_CONFIGURATION_CHANNEL error when you attempt to establish a connection between a configuration channel and any other channel.

DL50SW_ATTR_IS_DEBOUNCED
This attribute indicates whether the entire switch module has settled since the last switching command. A value of VI_TRUE indicates that all signals going through the switch module are valid.

DL50SW_ATTR_IS_SOURCE_CHANNEL
This channel-based attribute specifies whether you want to identify the channel as a source channel. Typically, you set this attribute to VI_TRUE when you attach the channel to a power supply, a function generator, or an active measurement point on the unit under test, and you do not want to

connect the channel to another source. The driver prevents source channels from connecting to each other. The dl50Sw_Connect function returns the DL50SW_ERROR_ATTEMPT_TO_CONNECT_SOURCES when you attempt to connect two channels that you identify as source channels.

DL50SW_ATTR_LOGICAL_NAME
A string containing the logical name you specified when opening the current IVI session. You may pass a logical name to the dl50Sw_init or dl50Sw_ InitWithOptions functions. The IVI Configuration utility must contain an entry for the logical name. The logical name entry refers to a virtual instrument section in the IVI Configuration file. The virtual instrument section specifies a physical device and initial user options.

DL50SW_ATTR_MANUFACTURER_ID
Returns the manufacturer identification number of the device. The instrument driver gets the value of this attribute when you pass VI_TRUE for the ID Query parameter to the dl50Sw_init or dl50Sw_InitWithOptions function.

DL50SW_ATTR_MAX_AC_VOLTAGE
This channel-based attribute returns the maximum AC voltage the channel can switch. The units are volts RMS.

DL50SW_ATTR_MAX_CARRY_AC_CURRENT
This channel-based attribute returns the maximum AC current the channel can carry. The units are amperes RMS.

DL50SW_ATTR_MAX_CARRY_AC_POWER
This channel-based attribute returns the maximum AC power the channel can carry. The units are volt-amperes.

DL50SW_ATTR_MAX_CARRY_DC_CURRENT
This channel-based attribute returns the maximum DC current the channel can carry. The units are amperes.

DL50SW_ATTR_MAX_CARRY_DC_POWER
This channel-based attribute returns the maximum DC power the channel can carry. The units are watts.

DL50SW_ATTR_MAX_DC_VOLTAGE
This channel-based attribute returns the maximum DC voltage the channel can switch. The units are volts.

DL50SW_ATTR_MAX_SWITCHING_AC_CURRENT
This channel-based attribute returns the maximum AC current the channel can switch. The units are amperes RMS.

DL50SW_ATTR_MAX_SWITCHING_AC_POWER
This channel-based attribute returns the maximum AC power the channel can switch. The units are volt-amperes.

DL50SW_ATTR_MAX_SWITCHING_DC_CURRENT
This channel-based attribute returns the maximum DC current the channel can switch. The units are amperes.

DL50SW_ATTR_MAX_SWITCHING_DC_POWER
This channel-based attribute returns the maximum DC power the channel can switch. The units are watts.

DL50SW_ATTR_MODEL_CODE
Returns the model code for the device. The instrument driver gets the value of this attribute when you pass VI_TRUE for the ID Query parameter to the dl50Sw_init or dl50Sw_InitWithOptions function.

DL50SW_ATTR_NUM_OF_ANALOG
This attribute returns the number of relays attached to the analog system bus backplane.

DL50SW_ATTR_NUM_OF_COLUMNS
This attribute returns the number of columns of a matrix or scanner. If the switch module is a scanner, this value is the number of input channels. The DL50SW_ATTR_WIRE_MODE attribute affects the number of available columns. For example, if your module has 8 input lines and you use the four-wire mode, then the number of columns you have available is 2.

DL50SW_ATTR_NUM_OF_DIGITAL
This attribute returns the number of relays that attach the digital pass through channels to the analog system bus backplane.

DL50SW_ATTR_NUM_OF_ROWS
This attribute returns the number of rows of a matrix or scanner. If the switch module is a scanner, this value is the number of output channels. The DL50SW_ATTR_WIRE_MODE attribute affects the number of available rows. For example, if your module has 2 output lines and you use the two-wire mode, then the number of rows you have available is 1.

DL50SW_ATTR_QUERY_INSTRUMENT_STATUS
Specifies whether the instrument driver queries the instrument status after each operation. Querying the instrument status is very useful for debugging. After you validate your program, you can set this attribute to VI_FALSE to disable status checking and maximize performance The instrument driver can choose to ignore status checking for particular attributes regardless of the setting of this attribute. The default value is VI_FALSE. Use the dl50Sw_ InitWithOptions function to override this value.

DL50SW_ATTR_RANGE_CHECK
Specifies whether to validate attribute values and function parameters. If enabled, the instrument driver validates the parameter values that you pass to driver functions. Range checking parameters is very useful for debugging. After you validate your program, you can set this attribute to VI_FALSE to disable range checking and maximize performance. The default value is VI_TRUE. Use the dl50Sw_InitWithOptions function to override this value.

DL50SW_ATTR_RECORD_COERCIONS
Specifies whether the IVI engine keeps a list of the value coercions it makes for integer and real type attributes. You call dl50Sw_GetNextCoercionRecord to extract and delete the oldest coercion record from the list. The default value is VI_FALSE. Use the dl50Sw_InitWithOptions function to override this value.

DL50SW_ATTR_RELAY_COUNT
This attribute returns the number of relays on this card.

DL50SW_ATTR_SETTLING_TIME
This channel-based attribute returns the maximum length of time from after you make a connection until the signal flowing through the channel settles. The units are seconds.

DL50SW_ATTR_SIMULATE
Specifies whether or not to simulate instrument driver I/O operations. If simulation is enabled, instrument driver functions perform range checking and call Ivi_GetAttribute and Ivi_SetAttribute functions, but they do not perform instrument I/O. For output parameters that represent instrument data, the instrument driver functions return calculated values. The default value is VI_FALSE. Use the dl50Sw_InitWithOptions function to override this value.

DL50SW_ATTR_SLOT_NUM
The slot number of the Resource Interface Chassis where the 2050 RIC board is located.

DL50SW_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION
The major version number of the class specification with which this driver is compliant.

DL50SW_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION
The minor version number of the class specification with which this driver is compliant.

DL50SW_ATTR_SPECIFIC_DRIVER_DESCRIPTION
A string that contains a brief description of the specific driver.

DL50SW_ATTR_SPECIFIC_DRIVER_PREFIX
A string that contains the prefix for the instrument driver. The name of each user-callable function in this driver starts with this prefix.

DL50SW_ATTR_SPECIFIC_DRIVER_REVISION
A string that contains additional version information about this instrument driver.

DL50SW_ATTR_SPECIFIC_DRIVER_VENDOR
A string that contains the name of the vendor that supplies this driver.

DL50SW_ATTR_SUPPORTED_INSTRUMENT_MODELS
Contains a model code of the instrument. For drivers that support more than one device, this attribute contains a comma-separated list of supported instrument models.

DL50SW_ATTR_WIRE_MODE

This attribute specifies the wire mode of the switch module. This attribute affects the values of the DL50SW_ATTR_NUM_OF_ROWS and DL50SW_ATTR_NUM_OF_COLUMNS attributes. The actual number of input and output lines on the switch module is fixed, but the number of channels depends on how many lines constitute each channel.

Selftest

Discrete Switching

Overview

The 0050-1218A Discrete Switch Pass-through Selftest board was designed to evaluate the 0000-6622 PXI Discrete Switch board attached to Pass-through lines routed through the RIC prototyping card 0050-1592 or equivalent. The 6622 board is used to simulate dirty or "leaky" switch contacts. Usually, the value of the resistors is 47 Ohm and 100K Ohm. However, these values can be project specific, as in the case of the GMT 900. The values here are 20 Ohm and 100K respectively. It should be noted that there is a posistor in series with the 20 Ohm resistor for current limiting purposes. The selftest for this board uses a 4-wire test to measure the 20 Ohm resistor and, although the 4-wire is still connected, for the 100K. There are 4 banks of 8 channels. Each bank of 8 channels has two busses associated with it so that an external positive or negative voltage can be applied during normal use, i.e.

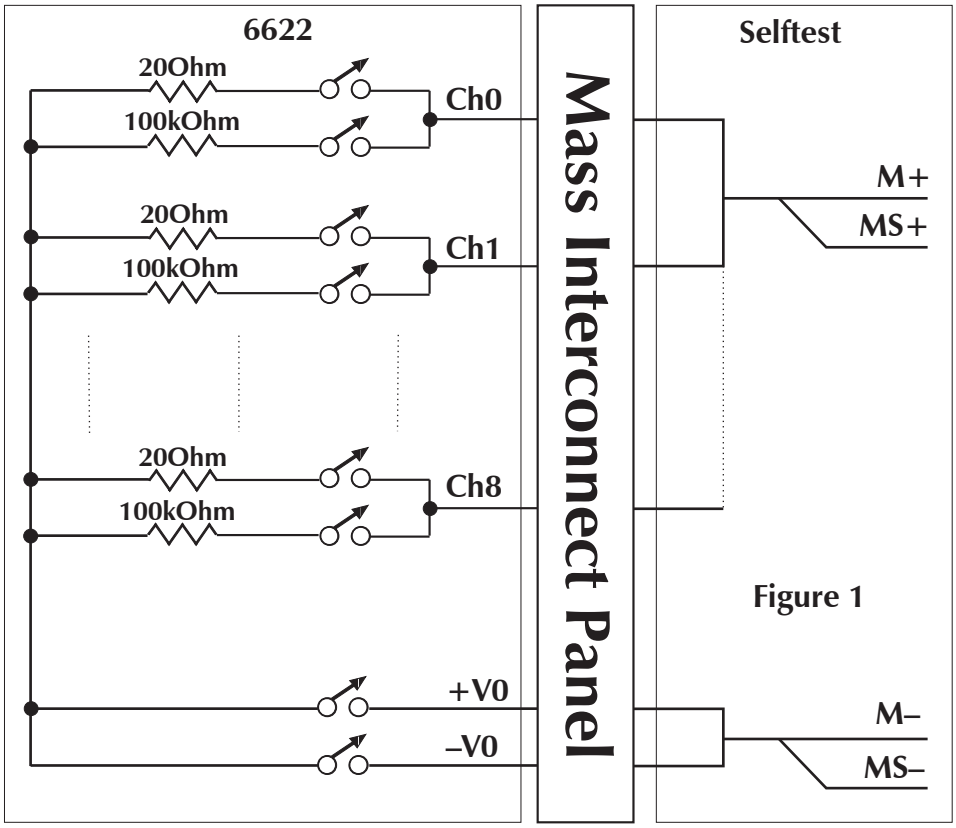


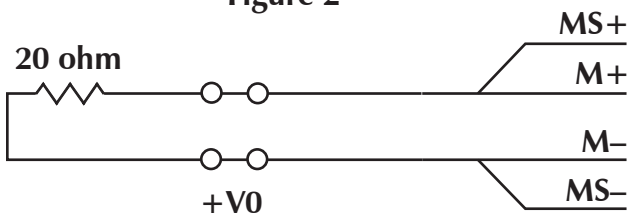
Figure 1

CH0-CH7 have a "+V0" bus and a "–V0 bus" @ Bank 0
CH8-CH15 have a "+V1" bus and a "–V1 bus" @ Bank 1
CH16-CH23 have a "+V2" bus and a "–V2 bus" @ Bank 2
CH24-CH31 have a "+V3" bus and a "–V3 bus" @ Bank 3

One bank of 8 channels is shown in figure 1. All the switching for these tests is done on the 6622 using IVI calls. The meter is utilised in 4-wire measurement mode. The tests are conducted as follows:

For the first test, CH0 is connected to the 20 Ohm resistor and the +V0 terminal is switched in on the 6622 card (Figure 2).

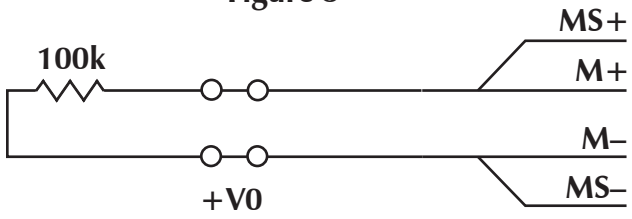
Figure 2



Due to the series posistor, not shown, and under normal ambient operating temperatures, the expected returned reading should be in the range 22 to 30 Ohms. If for some reason the 6622 board gets overheated this test will fail with a higher resistance. One possible reason for this board overheating could be the failure of the fans located below the PXI rack.

Next, CH0 is connected to the 100K resistor, instead of the 20 Ohm resistor. The +V0 terminal remains switched on the 6622 card (Figure 3).

Figure 3



The third test disconnects the +V0 bus and connects the –V0 bus. The 20 ohm resistor is connected to this bus and measured.

For the fourth test, CH0 is connected to the 100K resistor, instead of the 20 Ohm resistor. The –V0 terminal remains switched on the 6622 card.

The four tests described above are then repeated for each channel.

MRly Selftest

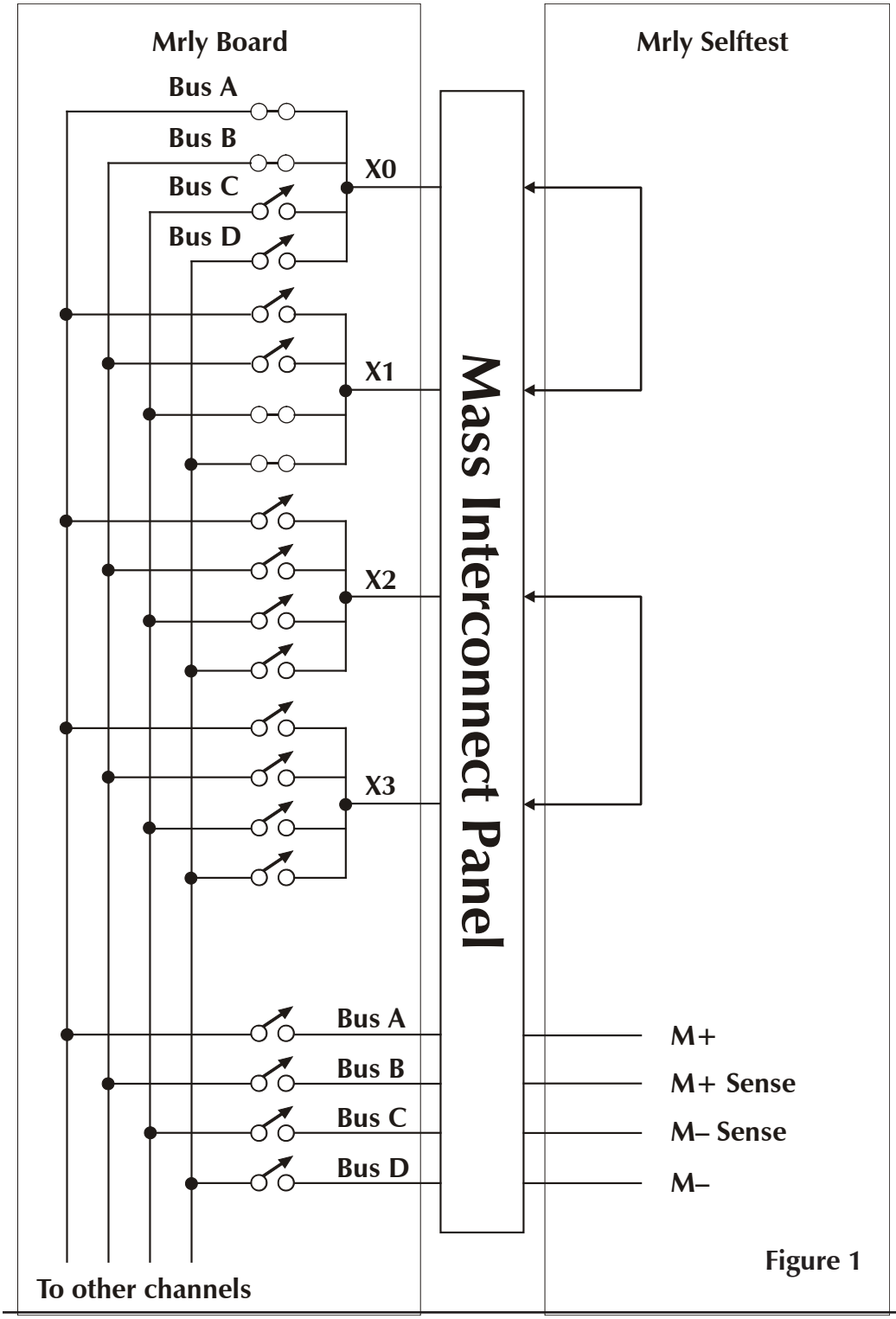

Mrly Board
Mrly Selftest
Bus A
Bus B
Bus C
Bus D
X0
X1
X2
X3
Mass Interconnect Panel
Bus A
Bus B
Bus C
Bus D
M+
M+ Sense
M– Sense
M–
To other channels

Figure 1

Overview

The 0050-1204A Matrix Relay Selftest board was designed to evaluate the 0050-1536 MRLY board and minimally test PXI resources attached to its Pass-through lines. There are two tests which are run on the MRLY cards. The first uses a 4-wire measurement to obtain the resistance between 4 relays and signal paths. The second uses a 2-wire measurement which is essentially a "go, no go" test.

MRly Test #1
Inside the Selftest assembly, adjacent MRLY channels are connected together in pairs, i.e. X0 is directly connected to X1 via a straight wire loop, X2 to X3, …….. X126 to X127. The four MRLY buses are then routed to the NI 4070 measurement card on the 4 "meter leads", M+, MS+, MS– and M–.

As can be seen in Figure 1, the first test closes the MRLY relays ready for the 4-wire test. The meter probes are connected as shown to the left.

X0 to BUSA & Meter Plus Lead	(M+)
X0 to BUSB & Meter Plus Sense Lead	(MS+)
X1 to BUSC & Meter Return Sense Lead	(MS–)
X1 to BUSD & Meter Return Lead	(M–)

This test measures the resistance, not only of the relays selected on the different channels, but also the resistance paths through the MRLY board and the Virginia Panel Mass Interconnect system. The measured resistance is hence much higher than one would expect from the relay contacts alone. The expected resistance value is in the range 0.1 to 0.8 Ohms. The actual reading for this first test is stored in Teststand in the array:

"FileGlobals.CLOSED_MRLY_MEASUREMENT_RESULTS_ARRAY[0]"

The second test switches the relays as follows:

Once again a 4-wire measurement is taken at the 4 buses; this time they are connected as shown in Figure 2:

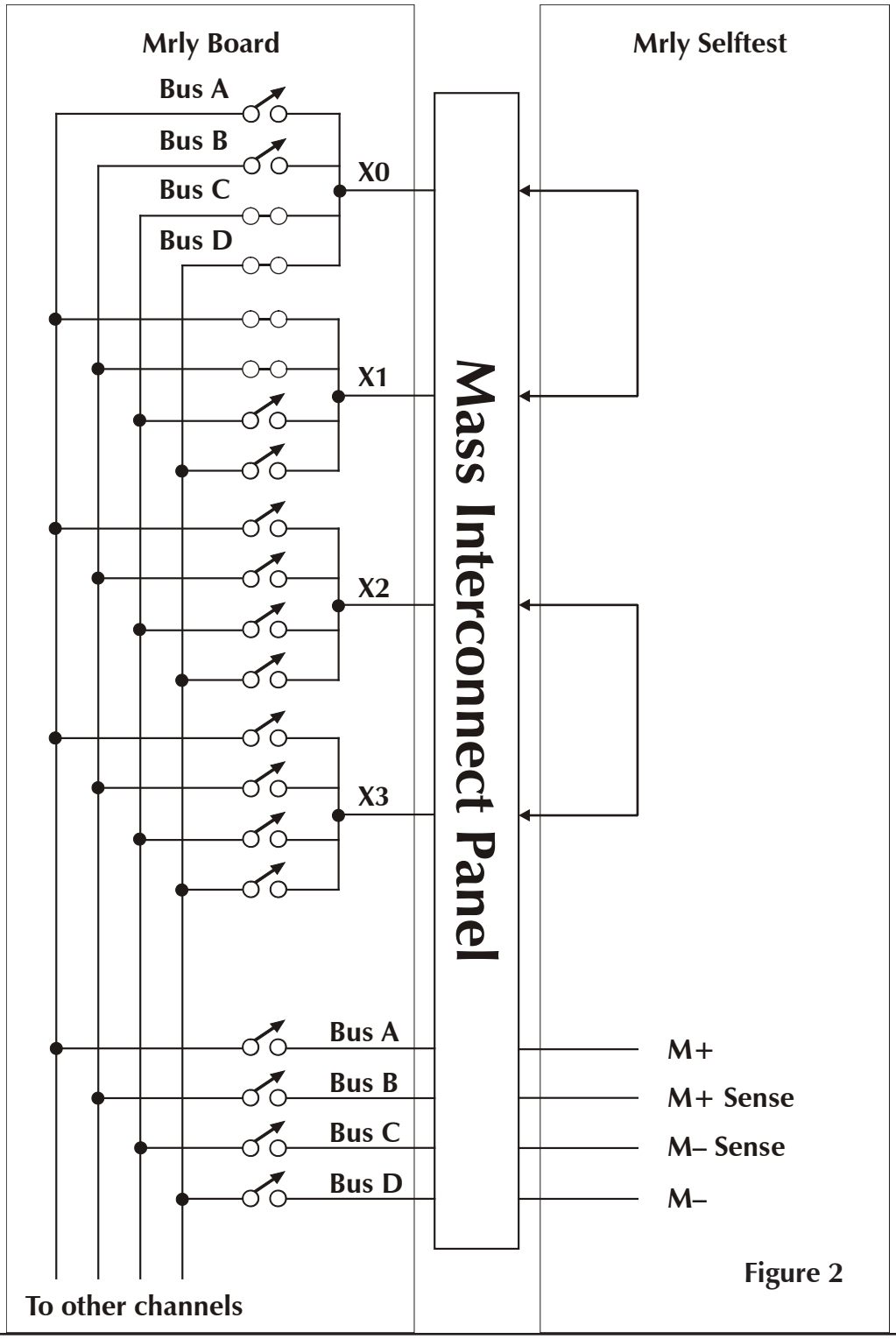

Mrly Board
Mrly Selftest
Bus A
Bus B
Bus C
Bus D
X0
X1
X2
X3
Mass Interconnect Panel
Bus A
Bus B
Bus C
Bus D
M+
M+ Sense
M– Sense
M–
To other channels

Figure 2

X0 to BUSC & Meter Plus Lead	(M+)
X0 to BUSD & Meter Plus Sense Lead	(MS+)
X1 to BUSA & Meter Return Sense Lead	(MS–)
X1 to BUSB & Meter Return Lead	(M–)

The actual reading for this second test is stored in TestStand in the array:

"FileGlobals.CLOSED_MRLY_MEASUREMENT_RESULTS_ARRAY[1]"

The rest of the tests in MRLY test 1 follow the same pattern as above for each channel pair i.e:

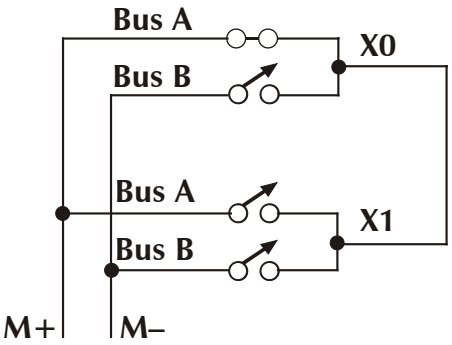
TEST #	CONNECTIONS	MEASUREMENT ARRAY INDEX
3	X2 to BUS0 & M+ X2 to BUS1 & MS+ X3 to BUS2 & MS- X3 to BUS3 & M-	FileGlobals.CLOSED_MRLY_MEASUREMENT_RESULTS_ARRAY[2]
4	X2 to BUS3 & M+ X2 to BUS2 & MS+ X3 to BUS0 & MS- X3 to BUS1 & M-	FileGlobals.CLOSED_MRLY_MEASUREMENT_RESULTS_ARRAY[3]
:	:	:
:	:	:
:	:	:
127	X126 to BUS0 & M+ X126 to BUS1 & MS+ X127 to BUS2 & MS- X127 to BUS3 & M-	FileGlobals.CLOSED_MRLY_MEASUREMENT_RESULTS_ARRAY[126]
128	X126 to BUS0 & M+ X126 to BUS1 & MS+ X127 to BUS2 & MS- X127 to BUS3 & M-	FileGlobals.CLOSED_MRLY_MEASUREMENT_RESULTS_ARRAY[127]

MRLY Test 2

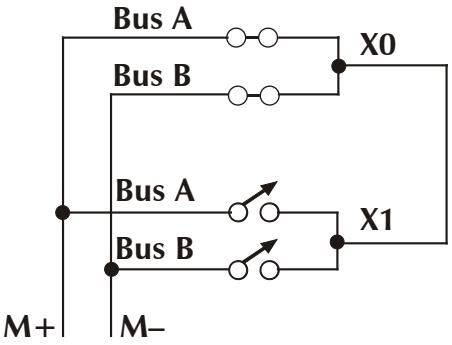
This test utilizes the NI 4070 meter card in 2-wire Mode. The two meter leads are routed to the NI 4070 measurement card in such a way that they be connected to any bus through internal switching on the selftest card.

The result of each test is passed to Teststand into the variable **fileGlobals.STATE[0]**. A successful test is indicated by a "1" returned to this array. Any other number suggests a problem within the group of four relays being tested. This test will pick up shorts which are not detected in the first MRLY tests.

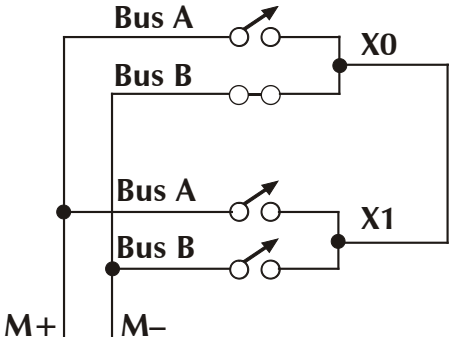
Each test is broken down into three smaller tests. The illustration below is for channel 0 only, but the same tests are repeated for each of the other 128 channels:



Shorts Test #1
Open expected



Shorts Test #2
Short expected



Shorts Test #3
Open expected

If either the first or third test fails, indicating one or more shorted relays, then the number "2" is passed back to TestStand. If the second test fails, indicating an open circuit in one of the two relays being tested on that channel, a "4" is passed back to TestStand.

When the relays on busses A and B have been tested on all 128 channels, the same tests are repeated on busses C and D for the 128 channels.

These tests, and all subsequent tests, are dependant on a number of variables:

1. Power to the selftest unit is established. If the supply to the selftest is off, or faulted, then obviously no tests will work. Power to the selftest unit is established through the connector in RIC slot 1. (Unfortunately, the RIC slots and the selftest slots use a different number base. Slot 1 on the RIC is slot 0 on the selftest unit). A quick check to see if the supply is on, this designated DEV12 in software, is to physically look at the supply controller on the back of the tester. It should have 3 green LED's lit on the right and a further 3 green LED's on the left. Any red LED's indicate a fault has occurred and will have to be reset. Note also, at this point, that the RIC supply controller, DEV 15, can be observed for faults too.

2. Communications between the selftest unit and the system is via a USB port. The USB port is "seen" as a serial COM port by the system. This is achieved with a software driver. It is important to note that, after system booting, one must wait for a period of 5 minutes before communications with the selftest are established. The reason for this is that the NI software takes precedence on boot-up and takes a few minutes to do it's internal setup and checks. Of course this should not present a problem in the

normal situation since the system should warmed up for 30 minutes before conducting selftest.

3. All cards in the Resource Interconnect Controller (RIC) chassis MUST be screwed into position when being seated otherwise intermittent failures WILL be obtained.

4. For this, and all subsequent tests, the path for the signals back to the NI 4070 card must be preserved. For the GMT 900 selftest, the NI 4070 card is situated in PXI slot 3. The meter probes are passed to the selftest board via a pass-through card located above the 4070 onto the Mass Interconnect Connector (MIC). The “meter leads” appear on the pins A48 (M+), B48 (MS+), C48 (MS-) and D48 (M-). Looking at the MIC from the front of the tester the connections are, left to right from the top, A1, B1, C1 & D1 (A48 is the bottom left connection). A quick test to see if these meter leads are reaching the MIC is to take off the selftest unit, connect a known resistor into the MIC, on the correct pins, and then open the NI Meter front panel software. If the corrections are correct then the panel should give the resistor reading.

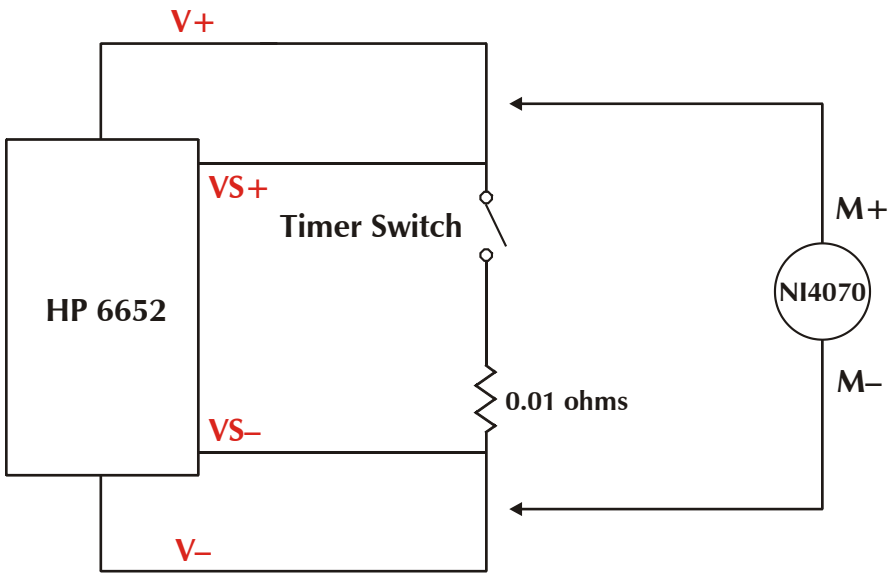
PPS Selftest

Overview

The 0050-1216A PPS Selftest board was designed to evaluate tester capabilities relative to power supply resources ported through the 0050-5104 PPS/MIC interconnect. The HP 6652 power supply is capable of delivering 0-20V at 0-25A. The purpose of these tests is to ensure the supply can deliver the voltages and currents specified.

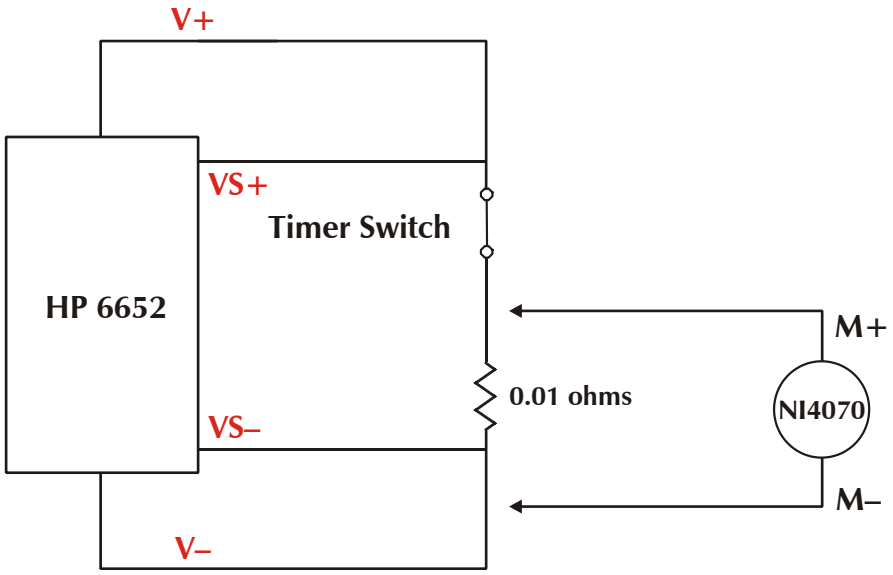
Test 1
The first batch of tests cycles the HP 6652 through it's range of voltages, 0-20V. The voltage is measured each time the voltage of the supply is re-programmed. This test exercises the capability of the power supply to provide the proper voltage over its full-scale range.

Simplified diagram of connections for the voltage tests:



Test 2
For the current tests the meter is connected across the 0.01 Ohms resistor. The current is calculated using Ohms law. This test exercises the capability of the power supply to provide current over a range.

Simplified diagram of connections for the current tests:



The *timer switch is connected to a timing circuit which is triggered just before the tests begin. Should a problem occur on the system, and it "hangs" during the middle of a test where the current may be a maximum, the circuit only remains active for a maximum of approximately 30 seconds. Thus, the circuit is protected from overheating and any related problems.

The signal path for these tests is as follows;

The outputs from the power supply are connected directly through RIC slot 1 and Selftest slot 0 (where ALL power comes into the Selftest Assembly). The power supply is programmed to the selected voltages and currents and measurements are read back through the meter.

If, for some reason, communications are lost or the ±15V or +5V supplies are lost, the timer switch will remain open, disconnecting all loads.

2050 Error Messages

Digalog 2050 Error Messages

The 2050 error codes are derived in the same way as with previous versions of tester, 2030 and 2040. i.e. **group number: item number**

The "group" number is found by dividing the error code by 256. The decimals are ignored to leave the group number.

The item number, which is the number shown after the colon, is derived by taking the group number and multiplying it by 256. This number is then subtracted from the original error code to give the item number.

Mathematically: **group number = integer part of (Error Code / 256)**
Item number = Error Code – (group number * 256)

Example: For the error code: 28260

The group number = integer part of (28260 / 256) = 110
The item number = 28260 – (110 * 256) = 100
Therefore 28260 = 110:100

2050 ERROR MESSAGES

98:001 (GPIB) Function requires GPIB to be Controller-In-Change.
98:002 (GPIB) Write function detected no Listeners.
98:003 (GPIB) GPIB board is not addressed correctly.
98:004 (GPIB) Invalid argument to function call.
98:005 (GPIB) GPIB board not System Controller.
98:006 (GPIB) I/O operation aborted.
98:007 (GPIB) Non-existent GPIB board.
98:008 (GPIB) Virtual DMA device error.
98:010 (GPIB) I/O started before previous operation completed.
98:011 (GPIB) No capability for operation.
98:012 (GPIB) File system error.
98:014 (GPIB) Command error during device call.
98:015 (GPIB) Serial poll status byte(s) lost.
98:016 (GPIB) SRQ stuck in the ON position.
98:020 (GPIB) Table problem.
98:021 (GPIB) Address or board is locked.
98:022 (GPIB) The GPIB-ENET was already on-line, and the default board configuration sent to it differs from than configuration under which it was already operating. This is only a warning. The board configuration has not been changed, but the operation has otherwise completed successfully.
98:023 (GPIB) The GPIB library was not linked. Dummy functions were linked instead.
98:024 (GPIB) Error loading GPIB-32.DLL. The MS Windows error code is in ibcnt.
98:025 (GPIB) Unable to find the function in GPIB-32.DLL. The MS Windows error code is in ibcnt.
98:026 (GPIB) Unable to find globals in GPIB-32.DLL. The MS Windows error code is in ibcnt .
98:027 (GPIB) Not a National Instruments GPIB-32.DLL.
98:028 (GPIB) Unable to acquire Mutex for loading DLL. The MS Windows error code is in ibcnt.
98:029 (GPIB) Unable to register callback function with MS Windows.
98:030 (GPIB) The callback table is full.

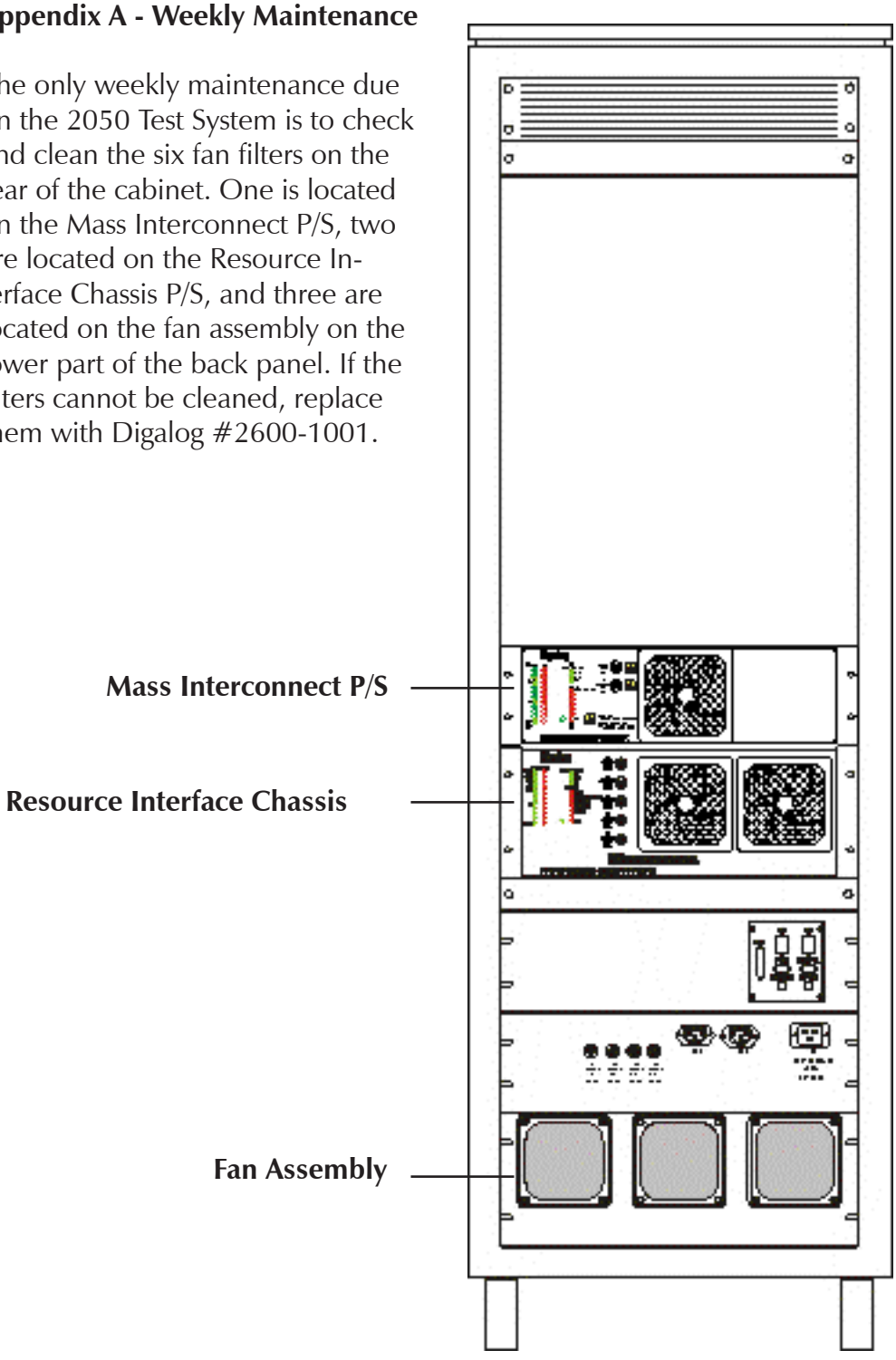
110:100 (RS232) Unknown system error.
110:101 (RS232) Invalid port number.
110:102 (RS232) Port is not open.
110:103 (RS232) Unknown IO error.
110:104 (RS232) Unexpected internal error.
110:105 (RS232) No serial port found.
110:106 (RS232) Cannot open port.
110:107 (RS232) Memory allocation error.
110:108 (RS232) Unable to allocate system resources.
110:109 (RS232) Invalid parameter.
110:110 (RS232) Invalid baud rate.
110:111 (RS232) Invalid parity mode.
110:112 (RS232) Illegal Number of data bits.

110:113	(RS232)	Illegal number of stop bits.
110:114	(RS232)	Bad file handle.
110:115	(RS232)	Error in performing file I/O.
110:116	(RS232)	Invalid count - must be 0 or greater.
110:117	(RS232)	Invalid interrupt level.
110:118	(RS232)	Default port address not defined for ports 8 through 32.
110:119	(RS232)	I/O operation timed out.
110:120	(RS232)	Break time must be between 0 and 255.
110:121	(RS232)	Invalid input queue size.
110:122	(RS232)	Invalid output queue size.
110:123	(RS232)	General I/O error.
110:124	(RS232)	Buffer parameter is null.
110:125	(RS232)	No acknowledgement received after packet sent.
110:126	(RS232)	Packet not sent within retry limit.
110:127	(RS232)	Packet not received within retry limit.
110:128	(RS232)	Unexpected end of transmission received.
110:129	(RS232)	Unable to read packet number.
110:130	(RS232)	Packet number inconsistency.
110:131	(RS232)	Packet data could not be read.
110:132	(RS232)	Checksum could not be read.
110:133	(RS232)	Received checksum does not match computed checksum.
110:134	(RS232)	Invalid packet size.
110:135	(RS232)	Error opening file.
110:136	(RS232)	Error reading file.
110:137	(RS232)	Initial negative acknowledgment not received.
110:138	(RS232)	Acknowledgement not received following end of.
110:139	(RS232)	Error while writing tot file.
110:140	(RS232)	Did not receive start of data nor end of transmission when one was expected.
110:141	(RS232)	Transfer was cancelled because CAN byte was received.
110:142	(RS232)	Invalid start delay.
110:143	(RS232)	Invalid maximum number of tries.
110:144	(RS232)	Invalid wait period.
110:145	(RS232)	Invalid packet size.
110:146	(RS232)	Unable to read CRC.
110:147	(RS232)	CRC error.
110:150		The returned selftest ID from the backplane was invalid.
110:151		Invalid slot number parameter passed.
110:152		Invalid address parameter passed.
110:153		Invalid byte parameter passed.
110:154		Failed to readback the same data as written.
110:170	(IVI)	The specified termination character was read.
110:171	(IVI)	The specified number of bytes was read.
110:172	(IVI)	Reset not supported.
110:173	(IVI)	Miscellaneous or system error occurred.
110:174	(IVI)	Invalid session handle.
110:175	(IVI)	Timeout occurred before operation could complete.

110:176	(IVI)	Violation of raw write protocol occurred.
110:177	(IVI)	Violation of raw read protocol occurred.
110:178	(IVI)	Device reported an output protocol error.
110:179	(IVI)	Device reported an input protocol error.
110:180	(IVI)	Bus error occurred during transfer.
110:181	(IVI)	Invalid setup (attributes are not consistent).
110:182	(IVI)	A “no listeners” condition was detected.
110:183	(IVI)	This interface is not the controller-in-charge.
110:184	(IVI)	Operation is not supported on this session.
110:200	(IVI)	Unknown Generic IVI Error.

Appendix A - Weekly Maintenance

The only weekly maintenance due on the 2050 Test System is to check and clean the six fan filters on the rear of the cabinet. One is located on the Mass Interconnect P/S, two are located on the Resource Interface Chassis P/S, and three are located on the fan assembly on the lower part of the back panel. If the filters cannot be cleaned, replace them with Digalog #2600-1001.



Appendix B - Recommended Spare Parts

Qnty	Digalog P/N	Description
2	0001-0014A	Low Current Relay Board
10001-0064A		40A Relay Board, W/O Curr
12000-2026		Controller, USB TO GPIB, Windows 2000/XP
1	4100-1027	MXI-3 PCI/PXI KIT,PCI-8330,PXI-8330,2M
	CopperCable	
1	0050-1536A	Digalog 2050 Matrix relay board
10050-1574A		Digalog 2050, Bus control card
10050-1592A		Digalog 2050, Prototyping card
1	4100-0124	AVT Multiple protocol unit,
14100-1026		NI 4070 PXI card
2	0050-5020-FS	Spare fuse Kit for RIC Power Supply
20050-5021-FS		Spare fuse Kit for MIC Power Supply
10050-5010		Mass Interconnect Power Box

Appendix C - 2050 Export Prototyping Card Functions

InitProtoRelayControl

This routine returns a handle to the 2050 Prototyping Card specified by "ProtoDesc". This handle is then passed to the calls used to control the relays. Every handle obtained using this call must be freed by using the "CloseProtoRelayControl()" call. If the handle is not freed, eventually the system will run out of available handles or it will run out of memory.

A handle is obtained by passing in the VISA style descriptor string for the 2050 Prototyping Card. An example would be "2050::4::INSTR". Where the '4' stands for the R.I.C. slot number the card is in. R.I.C. slot numbers can be 1 to 18.

CVI Declaration:

u_int32 DLIprc_InitProtoRelayControl(char *ProtoDesc, u_int32 *ProtoHandle)

u_int32 DLIprc_InitProtoRelayControl(*ProtoDesc, *ProtoHandle);

***ProtoDesc**

Resource descriptor of the card.

***ProtoHandle**

Returns the obtained handle.

CloseProtoRelayControl

This routine frees a handle to the 2050 Prototyping card that was previously obtained. The handle is obtained by using the “InitProtoRelayControl()” call. Once the handle is freed it is not valid anymore. Every handle obtained using this call must be freed by using the “CloseProtoRelayControl()” call. If the handle is not freed, eventually the system will run out of available handles or it will run out of memory.

CVI Declaration:

u_int32 DLIprc_CloseProtoRelayControl(u_int32 *ProtoHandle)

u_int32 DLIprc_CloseProtoRelayControl(*ProtoHandle);

***ProtoHandle**

Handle of the R.I.C Prototyping Card to access.

ProtoRelayControl

This function is used to open or close one relay which is being controlled by the 2050 R.I.C Prototyping card given by 'ProtoHandle'. The 'ProtoHandle' parameter is the handle to the card. This handle is obtained by using the "Init-ProtoRelayControl()" call.

This call does not wait for the relay to open or close. It is up to the user to provide a delay long enough for the relay to open or close.

CVI Declaration:

u_int32 DLIprc_ProtoRelayControl(u_int32 ProtoHandle, int16 Relay, int16 State)

u_int32 DLIprc_ProtoRelayControl(ProtoHandle, Relay, State);

ProtoHandle

Handle of the R.I.C Prototyping Card to access.

Relay

Relay number to access
= 0 to 127

State

Specifies if the relay should be opened or closed
= 0 - Open relay
= 1 - Close relay

ProtoRelayControlList

This function is used to open or close a list of relays which is being controlled by the 2050 R.I.C Prototyping card given by 'ProtoHandle'. The 'ProtoHandle' parameter is the handle to the card. This handle is obtained by using the "InitProtoRelayControl()" call.

This call accepts two arrays. One array contains the relays to access and the other array contains the state of the relay. The elements in the arrays are a one-to-one match. Meaning the first element in the state array specifies the state of the relay given in the first element of the relay array. The call also accepts a parameter that specifies how many array elements are valid. Each array must have as many elements as is specified by the 'ListLen' parameter.

The relays are opened or closed one at a time in the order they are in the array. This call does not wait for the relay to open or close. It is up to the user to provide a delay long enough for the relay to open or close.

CVI Declaration:

u_int32 DLIprc_ProtoRelayControlList(u_int32 ProtoHandle, int16 *Relay, int16 *State, int16 ListLen)

u_int32 DLIprc_ProtoRelayControlList(ProtoHandle, *Relay, *State, ListLen);

ProtoHandle

Handle of the R.I.C Prototyping Card to access.

***Relay**

Array of relay numbers that are to be changed.

***State**

Array of relay states that are to be changed.

ListLen

Length of the 'Relay' and 'State' arrays. It is up to the user to guarantee that the arrays are as long as 'ListLen'.

ResetProtoControlRelays

This function is used to open all of the relays that are controlled from one R.I.C. Prototyping card.

CVI Declaration:

u_int32 DLIprc_ResetProtoControlRelays(u_int32 ProtoHandle)

u_int32 DLIprc_ResetProtoControlRelays(ProtoHandle);

ProtoHandle

Handle of the R.I.C Prototyping Card to access.

Digalog
Mass Interconnect Power Box
Mass Interconnect P/S Front
0050-0033 PXI Fan Assembly
Digalog
OK FAULT
CONTROLLER
-12
+12
+5
FAULTS
External Fault
+15
-15
+5
Temp
COP
System Controller
Digalog Signal Conditioning Boards
Analog Backplane
Digital Backplane
Mass Interconnect Panel
GEMINI
0050-2001
0050-2002
0050-2014 Prox Fault
Mass Interconnect P/S Rear
6000-1039 GPIB
0050-2012
Mass Interconnect Power
PXI Data Bus
0050-2010
0050-2006 24VDC
6000-1021
6000-1021
PXI Bus
0050-2005
0050-2013
RIC P/S Rear
6000-1039 GPIB
0050-2010 PBP/S Fault
0050-2011 RCS Fault
6000-1021
RIC P/S Front
Product PPS
6000-1039
6000-1038 GPIB
0000-3983
6000-1021
6000-1021
Product Power
0000-3984 - Fault/Inhibit
TB1
TB2
P1
P2
P3
P4
P5
RCS
PXI Bus Power
Resource Interface Chassis Power Box
0050-2003 UPS REPO Cable
0050-2011
UPS
0000-5934
0050-2006 24VDC
6000-1021
0000-5934
0050-5101 E-Stop On/Off Assembly
On - Off
E-Stop
0000-5928
208 - 240VAC In
0050-2007
0050-0035 Receiver Fan Assembly
0003-0006 AC Fan Assembly
F1 250VAC 4A MDL
F2 250VAC 1A MDL
F3 250VAC 10A MDL
F4 250VAC 10A MDL
J14
J15
J1 208-230VAC 10A 50-60Hz
J2
J3
J4
J5
J6
J7
J8
J9
J10
J11
J14
J15
0050-2006 24VDC
Front - AC Power Management - Rear